

Thyrix Reference Manual

1.1

Generated by Doxygen 1.3.9.1

Thu Sep 1 18:03:33 2005

Contents

1	Thyrix Hierarchical Index	1
1.1	Thyrix Class Hierarchy	1
2	Thyrix Class Index	3
2.1	Thyrix Class List	3
3	Thyrix File Index	5
3.1	Thyrix File List	5
4	Thyrix Class Documentation	9
4.1	ArticulatedAgentBase Class Reference	9
4.2	ArticulatedAgentQuasistatic Class Reference	14
4.3	ArticulatedComponent Class Reference	18
4.4	ArticulatedLimb Class Reference	22
4.5	ArticulatedLink Class Reference	26
4.6	Border Class Reference	34
4.7	CachingController Class Reference	37
4.8	CappedRectangle Class Reference	39
4.9	Circle Class Reference	45
4.10	Color Class Reference	50
4.11	ColorDefinitions Class Reference	52
4.12	ColorDefinitions::ltstr Struct Reference	53
4.13	ColorDefinitions::SimpleColor Struct Reference	54
4.14	ComposedPhysicalObject Class Reference	55
4.15	ContactInfo Class Reference	61
4.16	ContactSolver Class Reference	65
4.17	Controller Class Reference	71
4.18	ElasticLink Class Reference	73
4.19	Elastoid Class Reference	75

4.20 GlobalContactInfo Class Reference	79
4.21 Graph Class Reference	81
4.22 GraphData Class Reference	84
4.23 GraphicalObject Class Reference	86
4.24 GUI Class Reference	88
4.25 GUIWx Class Reference	94
4.26 Integrator Class Reference	98
4.27 Iunctus Class Reference	100
4.28 IunctusSimulator Class Reference	103
4.29 LinkContactInfo Class Reference	105
4.30 MathTools Class Reference	107
4.31 Matrix3 Class Reference	109
4.32 MTRandom Class Reference	112
4.33 Pac Class Reference	115
4.34 PacSimulator Class Reference	118
4.35 PhysicalObject Class Reference	119
4.36 Random Class Reference	130
4.37 RandomController Class Reference	132
4.38 Simulator Class Reference	133
4.39 Simulator::ObjectPair Class Reference	140
4.40 SimulatorThread Class Reference	142
4.41 Spherus Class Reference	146
4.42 SpherusSimulator Class Reference	153
4.43 SpikeGraph Class Reference	154
4.44 SymmetricMatrix3 Class Reference	157
4.45 SystemSolver Class Reference	159
4.46 ThyrixApplication Class Reference	161
4.47 ThyrixMainFrame Class Reference	162
4.48 ThyrixParameters Class Reference	168
4.49 Vector2 Class Reference	170
4.50 Vector3 Class Reference	176
4.51 VisualSensor Class Reference	180
4.52 World Class Reference	184
5 Thyrix File Documentation	187
5.1 ArticulatedAgentBase.cpp File Reference	187
5.2 ArticulatedAgentBase.h File Reference	188

5.3	ArticulatedAgentQuasistatic.cpp File Reference	189
5.4	ArticulatedAgentQuasistatic.h File Reference	190
5.5	ArticulatedComponent.cpp File Reference	191
5.6	ArticulatedComponent.h File Reference	192
5.7	ArticulatedLimb.cpp File Reference	193
5.8	ArticulatedLimb.h File Reference	194
5.9	ArticulatedLink.cpp File Reference	195
5.10	ArticulatedLink.h File Reference	196
5.11	Border.cpp File Reference	197
5.12	Border.h File Reference	198
5.13	CachingController.cpp File Reference	199
5.14	CachingController.h File Reference	200
5.15	CappedRectangle.cpp File Reference	201
5.16	CappedRectangle.h File Reference	202
5.17	Circle.cpp File Reference	203
5.18	Circle.h File Reference	204
5.19	Color.cpp File Reference	205
5.20	Color.h File Reference	206
5.21	ColorDefinitions.cpp File Reference	207
5.22	ColorDefinitions.h File Reference	208
5.23	ComposedPhysicalObject.cpp File Reference	209
5.24	ComposedPhysicalObject.h File Reference	210
5.25	ContactInfo.cpp File Reference	211
5.26	ContactInfo.h File Reference	212
5.27	ContactSolver.cpp File Reference	213
5.28	ContactSolver.h File Reference	214
5.29	Controller.cpp File Reference	215
5.30	Controller.h File Reference	216
5.31	ElasticLink.cpp File Reference	217
5.32	ElasticLink.h File Reference	218
5.33	Elastoid.cpp File Reference	219
5.34	Elastoid.h File Reference	220
5.35	GlobalContactInfo.h File Reference	221
5.36	Graph.cpp File Reference	222
5.37	Graph.h File Reference	223
5.38	GraphData.cpp File Reference	224

5.39 GraphData.h File Reference	225
5.40 GraphicalObject.cpp File Reference	226
5.41 GraphicalObject.h File Reference	227
5.42 GUI.cpp File Reference	228
5.43 GUI.h File Reference	229
5.44 GUIWx.cpp File Reference	230
5.45 GUIWx.h File Reference	231
5.46 Integrator.h File Reference	232
5.47 Iunctus.cpp File Reference	233
5.48 Iunctus.h File Reference	234
5.49 IunctusSimulator.cpp File Reference	235
5.50 IunctusSimulator.h File Reference	236
5.51 LinkContactInfo.cpp File Reference	237
5.52 LinkContactInfo.h File Reference	238
5.53 MathTools.cpp File Reference	239
5.54 MathTools.h File Reference	240
5.55 Matrix3.cpp File Reference	241
5.56 Matrix3.h File Reference	242
5.57 MTRandom.cpp File Reference	243
5.58 MTRandom.h File Reference	244
5.59 Pac.cpp File Reference	245
5.60 Pac.h File Reference	246
5.61 PacSimulator.cpp File Reference	247
5.62 PacSimulator.h File Reference	248
5.63 PhysicalObject.cpp File Reference	249
5.64 PhysicalObject.h File Reference	250
5.65 purgeContainer.h File Reference	251
5.66 Random.cpp File Reference	252
5.67 Random.h File Reference	253
5.68 RandomController.cpp File Reference	254
5.69 RandomController.h File Reference	255
5.70 resource.h File Reference	256
5.71 Simulator.cpp File Reference	257
5.72 Simulator.h File Reference	258
5.73 SimulatorThread.cpp File Reference	259
5.74 SimulatorThread.h File Reference	260

5.75 Spherus.cpp File Reference	261
5.76 Spherus.h File Reference	262
5.77 SpherusSimulator.cpp File Reference	263
5.78 SpherusSimulator.h File Reference	264
5.79 SpikeGraph.cpp File Reference	265
5.80 SpikeGraph.h File Reference	266
5.81 SymmetricMatrix3.cpp File Reference	267
5.82 SymmetricMatrix3.h File Reference	268
5.83 SystemSolver.cpp File Reference	269
5.84 SystemSolver.h File Reference	270
5.85 ThyrixApplication.cpp File Reference	271
5.86 ThyrixApplication.h File Reference	272
5.87 ThyrixMainFrame.cpp File Reference	273
5.88 ThyrixMainFrame.h File Reference	274
5.89 ThyrixParameters.cpp File Reference	275
5.90 ThyrixParameters.h File Reference	276
5.91 Vector2.cpp File Reference	278
5.92 Vector2.h File Reference	279
5.93 Vector3.cpp File Reference	280
5.94 Vector3.h File Reference	281
5.95 VisualSensor.cpp File Reference	282
5.96 VisualSensor.h File Reference	283
5.97 World.cpp File Reference	284
5.98 World.h File Reference	285
5.99 wxIncludes.h File Reference	286

Chapter 1

Thyrix Hierarchical Index

1.1 Thyrix Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ArticulatedLimb	22
Color	50
ColorDefinitions	52
ColorDefinitions::ltstr	53
ColorDefinitions::SimpleColor	54
ContactInfo	61
ContactSolver	65
Controller	71
CachingController	37
RandomController	132
ElasticLink	73
GlobalContactInfo	79
Graph	81
GraphData	84
GraphicalObject	86
PhysicalObject	119
Border	34
CappedRectangle	39
Circle	45
ComposedPhysicalObject	55
ArticulatedComponent	18
ArticulatedAgentBase	9
ArticulatedAgentQuasistatic	14
Iunctus	100
ArticulatedLink	26
Elastoid	75
Pac	115
Spherus	146
VisualSensor	180
GUI	88
GUIWx	94
Integrator	98

LinkContactInfo	105
MathTools	107
Matrix3	109
SymmetricMatrix3	157
MTRandom	112
Random	130
Simulator::ObjectPair	140
SimulatorThread	142
SpikeGraph	154
SystemSolver	159
ThyrixApplication	161
ThyrixMainFrame	162
ThyrixParameters	168
Vector2	170
Vector3	176
World	184
Simulator	133
IunctusSimulator	103
PacSimulator	118
SpherusSimulator	153

Chapter 2

Thyrix Class Index

2.1 Thyrix Class List

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

ArticulatedAgentBase	9
ArticulatedAgentQuasistatic	14
ArticulatedComponent	18
ArticulatedLimb	22
ArticulatedLink	26
Border	34
CachingController	37
CappedRectangle	39
Circle	45
Color	50
ColorDefinitions	52
ColorDefinitions::ltstr	53
ColorDefinitions::SimpleColor	54
ComposedPhysicalObject	55
ContactInfo	61
ContactSolver	65
Controller	71
ElasticLink	73
Elastoid	75
GlobalContactInfo	79
Graph	81
GraphData	84
GraphicalObject	86
GUI	88
GUIWx	94
Integrator	98
Iunctus	100
IunctusSimulator	103
LinkContactInfo	105
MathTools	107
Matrix3	109
MTRandom	112
Pac	115

PacSimulator	118
PhysicalObject	119
Random	130
RandomController	132
Simulator	133
Simulator::ObjectPair	140
SimulatorThread	142
Spherus	146
SpherusSimulator	153
SpikeGraph	154
SymmetricMatrix3	157
SystemSolver	159
ThyrixApplication	161
ThyrixMainFrame	162
ThyrixParameters	168
Vector2	170
Vector3	176
VisualSensor	180
World	184

Chapter 3

Thyrix File Index

3.1 Thyrix File List

Here is a list of all files with brief descriptions:

ArticulatedAgentBase.cpp	187
ArticulatedAgentBase.h	188
ArticulatedAgentQuasistatic.cpp	189
ArticulatedAgentQuasistatic.h	190
ArticulatedComponent.cpp	191
ArticulatedComponent.h	192
ArticulatedLimb.cpp	193
ArticulatedLimb.h	194
ArticulatedLink.cpp	195
ArticulatedLink.h	196
Border.cpp	197
Border.h	198
CachingController.cpp	199
CachingController.h	200
CappedRectangle.cpp	201
CappedRectangle.h	202
Circle.cpp	203
Circle.h	204
Color.cpp	205
Color.h	206
ColorDefinitions.cpp	207
ColorDefinitions.h	208
ComposedPhysicalObject.cpp	209
ComposedPhysicalObject.h	210
ContactInfo.cpp	211
ContactInfo.h	212
ContactSolver.cpp	213
ContactSolver.h	214
Controller.cpp	215
Controller.h	216
ElasticLink.cpp	217
ElasticLink.h	218
Elastoid.cpp	219

Elastoid.h	220
GlobalContactInfo.h	221
Graph.cpp	222
Graph.h	223
GraphData.cpp	224
GraphData.h	225
GraphicalObject.cpp	226
GraphicalObject.h	227
GUI.cpp	228
GUI.h	229
GUIWx.cpp	230
GUIWx.h	231
Integrator.h	232
Iunctus.cpp	233
Iunctus.h	234
IunctusSimulator.cpp	235
IunctusSimulator.h	236
LinkContactInfo.cpp	237
LinkContactInfo.h	238
MathTools.cpp	239
MathTools.h	240
Matrix3.cpp	241
Matrix3.h	242
MTRandom.cpp	243
MTRandom.h	244
Pac.cpp	245
Pac.h	246
PacSimulator.cpp	247
PacSimulator.h	248
PhysicalObject.cpp	249
PhysicalObject.h	250
purgeContainer.h	251
Random.cpp	252
Random.h	253
RandomController.cpp	254
RandomController.h	255
resource.h	256
Simulator.cpp	257
Simulator.h	258
SimulatorThread.cpp	259
SimulatorThread.h	260
Spherus.cpp	261
Spherus.h	262
SpherusSimulator.cpp	263
SpherusSimulator.h	264
SpikeGraph.cpp	265
SpikeGraph.h	266
SymmetricMatrix3.cpp	267
SymmetricMatrix3.h	268
SystemSolver.cpp	269
SystemSolver.h	270
ThyrixApplication.cpp	271
ThyrixApplication.h	272
ThyrixMainFrame.cpp	273

ThyrixMainFrame.h	274
ThyrixParameters.cpp	275
ThyrixParameters.h	276
Vector2.cpp	278
Vector2.h	279
Vector3.cpp	280
Vector3.h	281
VisualSensor.cpp	282
VisualSensor.h	283
World.cpp	284
World.h	285
wxIncludes.h	286

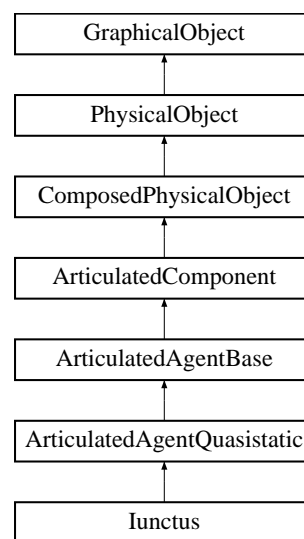
Chapter 4

Thyrix Class Documentation

4.1 ArticulatedAgentBase Class Reference

```
#include <ArticulatedAgentBase.h>
```

Inheritance diagram for ArticulatedAgentBase::



Public Member Functions

- [ArticulatedAgentBase](#) (std::string label="")
- virtual [~ArticulatedAgentBase](#) ()
- [ArticulatedLimb](#) * [addLimb](#) ()
- [ArticulatedLimb](#) * [addLimb](#) (real l, real theta)
- virtual void [registerPrimitives](#) ([Simulator](#) *simulator)
- virtual bool [detectContacts](#) ([PhysicalObject](#) *object, [GlobalContactInfoVector](#) *contacts)
- virtual void [detectInternalContacts](#) ([GlobalContactInfoVector](#) *contacts)
- virtual bool [detectMouseContact](#) (const [Vector2](#) &rMouse, [Vector2](#) &p, [PhysicalObject](#) *&object)
- virtual void [deleteContacts](#) ()

- void `fillContactMatrix` (`ContactSolver` *contactSolver, `ContactInfo` *contactInfo)
- virtual void `draw` (`GUI` *gui)
- virtual void `setOutlineColor` (`Color` color)
- virtual void `setFillColor` (`Color` color)

Protected Member Functions

- void `deleteLimbs` ()

Static Protected Member Functions

- void `solveSystem` (`SymmetricMatrix3` &m, const `Vector3` &c, `Vector3` &x)
- void `solveSystem` (const `SymmetricMatrix3` &m, const `real` determinant, const `Vector3` &c, `Vector3` &x)

Protected Attributes

- `ArticulatedLimbPVector` limbs

4.1.1 Detailed Description

Base class for articulated agents. Should be specialized for either classical mechanics or quasistatic mechanics, for different types of bodies.

Definition at line 19 of file `ArticulatedAgentBase.h`.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 `ArticulatedAgentBase::ArticulatedAgentBase (std::string label = "")`

sets the body to NULL.

Definition at line 38 of file `ArticulatedAgentBase.cpp`.

4.1.2.2 `ArticulatedAgentBase::~~ArticulatedAgentBase ()` [virtual]

Destructor; deletes the limbs.

Definition at line 45 of file `ArticulatedAgentBase.cpp`.

References `deleteLimbs()`.

4.1.3 Member Function Documentation

4.1.3.1 `ArticulatedLimb * ArticulatedAgentBase::addLimb (real l, real theta)`

Adds a limb to the agent, sets the properties of the limb and returns a reference to it. The limb is articulated at a point situated at `l` from the point of reference of the body, under and angle `theta`.

Definition at line 62 of file `ArticulatedAgentBase.cpp`.

References `addLimb()`, and `ArticulatedLimb::setProperties()`.

4.1.3.2 [ArticulatedLimb](#) * ArticulatedAgentBase::addLimb ()

Adds a limb to the agent and returns a reference to it.

Definition at line 50 of file ArticulatedAgentBase.cpp.

References [ArticulatedLimb::label](#), and [limbs](#).

Referenced by [addLimb\(\)](#), and [Iunctus::build\(\)](#).

4.1.3.3 void ArticulatedAgentBase::deleteContacts () [virtual]

Delete the contacts of the object.

Reimplemented from [PhysicalObject](#).

Reimplemented in [Iunctus](#).

Definition at line 173 of file ArticulatedAgentBase.cpp.

References [PhysicalObject::deleteContacts\(\)](#), [limbs](#), and [ArticulatedLimb::links](#).

Referenced by [Iunctus::deleteContacts\(\)](#).

4.1.3.4 void ArticulatedAgentBase::deleteLimbs () [protected]

Definition at line 208 of file ArticulatedAgentBase.cpp.

References [limbs](#), and [purgeContainer\(\)](#).

Referenced by [~ArticulatedAgentBase\(\)](#).

4.1.3.5 bool ArticulatedAgentBase::detectContacts ([PhysicalObject](#) * *object*, [GlobalContactInfoVector](#) * *contacts*) [virtual]

Contact handling.

Reimplemented from [ComposedPhysicalObject](#).

Definition at line 105 of file ArticulatedAgentBase.cpp.

References [ComposedPhysicalObject::detectContacts\(\)](#), [limbs](#), and [ArticulatedLimb::links](#).

4.1.3.6 void ArticulatedAgentBase::detectInternalContacts ([GlobalContactInfoVector](#) * *contacts*) [virtual]

Detects internal contacts. Does nothing for primitive objects or composed objects. To be overridden by complex objects that have parts that move relative to each other, such as articulated objects.

Reimplemented from [PhysicalObject](#).

Definition at line 119 of file ArticulatedAgentBase.cpp.

References [ComposedPhysicalObject::detectContacts\(\)](#), [limbs](#), and [ArticulatedLimb::links](#).

4.1.3.7 **bool ArticulatedAgentBase::detectMouseContact** (const **Vector2** & *rMouse*, **Vector2** & *p*, **PhysicalObject** *& *object*) [virtual]

Detects whether a click of the mouse at *rMouse* has touched the object. In case of contact, returns true and sets *p* to the relative vector between the center of the composed object and the contact point; *p* is expressed in the composed object reference frame because it rotates with the object while the object is dragged.

Reimplemented from [ComposedPhysicalObject](#).

Definition at line 159 of file `ArticulatedAgentBase.cpp`.

References `ComposedPhysicalObject::detectMouseContact()`, `limbs`, and `ArticulatedLimb::links`.

4.1.3.8 **void ArticulatedAgentBase::draw** (**GUI** * *gui*) [virtual]

Calls the draw methods for the body and the limbs.

Reimplemented from [ComposedPhysicalObject](#).

Reimplemented in [Iunctus](#).

Definition at line 212 of file `ArticulatedAgentBase.cpp`.

References `ComposedPhysicalObject::draw()`, and `limbs`.

Referenced by `Iunctus::draw()`.

4.1.3.9 **void ArticulatedAgentBase::fillContactMatrix** (**ContactSolver** * *contactSolver*, **ContactInfo** * *contactInfo*) [virtual]

Computes the contribution of the current object to the matrix of interactions between all objects that are in contact.

Reimplemented from [PhysicalObject](#).

Definition at line 186 of file `ArticulatedAgentBase.cpp`.

References `ContactInfo::alpha`, `ContactSolver::contactMatrix`, `ContactSolver::contactVector`, `Vector2::cross()`, `ContactInfo::n`, `ContactInfo::p`, `Vector2::rotate()`, `Vector2::setXY()`, and `ContactInfo::sigma`.

4.1.3.10 **void ArticulatedAgentBase::registerPrimitives** (**Simulator** * *simulator*) [virtual]

Registers all composing primitives of the object with the simulator.

Reimplemented from [ComposedPhysicalObject](#).

Definition at line 69 of file `ArticulatedAgentBase.cpp`.

References `limbs`, `ArticulatedLimb::links`, and `ComposedPhysicalObject::registerPrimitives()`.

4.1.3.11 **void ArticulatedAgentBase::setFillColor** (**Color** *color*) [virtual]

Sets fill color.

Reimplemented from [ComposedPhysicalObject](#).

Definition at line 228 of file `ArticulatedAgentBase.cpp`.

References `limbs`, and `ComposedPhysicalObject::setFillColor()`.

Referenced by `Iunctus::build()`.

4.1.3.12 `void ArticulatedAgentBase::setOutlineColor (Color color)` [virtual]

Sets outline color.

Reimplemented from `ComposedPhysicalObject`.

Definition at line 220 of file `ArticulatedAgentBase.cpp`.

References `limbs`, and `ComposedPhysicalObject::setOutlineColor()`.

Referenced by `Iunctus::build()`.

4.1.3.13 `void ArticulatedAgentBase::solveSystem (const SymmetricMatrix3 & m, const real determinant, const Vector3 & c, Vector3 & x)` [static, protected]

Solves the system of equations $m x = c$, given the precomputed determinant of m ; x is updated with the solution.

Definition at line 90 of file `ArticulatedAgentBase.cpp`.

References `Matrix3::getDeterminant()`, `Matrix3::setColumn()`, and `Vector3::setElement()`.

4.1.3.14 `void ArticulatedAgentBase::solveSystem (SymmetricMatrix3 & m, const Vector3 & c, Vector3 & x)` [static, protected]

Solves the system of equations $m x = c$; x is updated with the solution.

Definition at line 83 of file `ArticulatedAgentBase.cpp`.

References `SymmetricMatrix3::getDeterminant()`, `SymmetricMatrix3::mirror()`, and `real`.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, and `ArticulatedAgentQuasistatic::computeBodyDerivativesWithoutContacts()`.

4.1.4 Member Data Documentation

4.1.4.1 `ArticulatedLimbPVector ArticulatedAgentBase::limbs` [protected]

A list containing the limbs. The list owns the limbs.

Definition at line 61 of file `ArticulatedAgentBase.h`.

Referenced by `addLimb()`, `deleteContacts()`, `deleteLimbs()`, `detectContacts()`, `detectInternalContacts()`, `detectMouseContact()`, `draw()`, `registerPrimitives()`, `setFillColor()`, and `setOutlineColor()`.

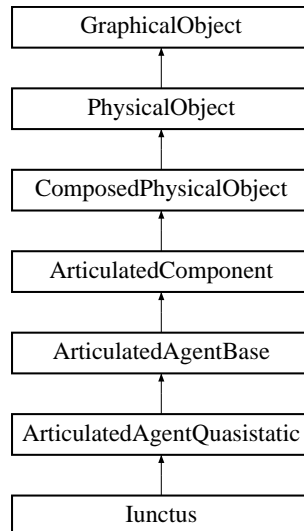
The documentation for this class was generated from the following files:

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

4.2 ArticulatedAgentQuasistatic Class Reference

```
#include <ArticulatedAgentQuasistatic.h>
```

Inheritance diagram for ArticulatedAgentQuasistatic::



Public Member Functions

- [ArticulatedAgentQuasistatic](#) (std::string [label](#)="")
- virtual [~ArticulatedAgentQuasistatic](#) ()

Protected Member Functions

- void [forwardKinematics](#) ()
- void [backwardDynamics](#) ()
- void [forwardAccelerations](#) ([ContactSolver](#) *contactSolver)
- virtual void [computeDerivativesWithoutContacts](#) ([ContactSolver](#) *contactSolver)
- virtual void [computeBodyDerivativesWithoutContacts](#) (real determinant)
- virtual void [computeDerivatives](#) ([GlobalContactInfoVector](#) *globalContacts)
- virtual void [computeBodyDerivatives](#) ([GlobalContactInfoVector](#) *globalContacts)
- void [computeForces](#) ([GlobalContactInfoVector](#) *globalContacts)
- void [computeTotalForces](#) ([GlobalContactInfoVector](#) *globalContacts)
- virtual void [integrate](#) (const [Integrator](#) &integrator)
- virtual void [rollback](#) ()

4.2.1 Detailed Description

Computes the velocities of the coordinates for an articulated agent, given forces, using a Featherstone-type algorithm, for the case of quasistatic mechanics.

Definition at line 14 of file [ArticulatedAgentQuasistatic.h](#).

4.2.2 Constructor & Destructor Documentation

4.2.2.1 ArticulatedAgentQuasistatic::ArticulatedAgentQuasistatic (std::string *label* = "")

Default constructor; sets the simulator.

Definition at line 19 of file ArticulatedAgentQuasistatic.cpp.

4.2.2.2 ArticulatedAgentQuasistatic::~~ArticulatedAgentQuasistatic () [virtual]

Destructor; does nothing.

Definition at line 23 of file ArticulatedAgentQuasistatic.cpp.

4.2.3 Member Function Documentation

4.2.3.1 void ArticulatedAgentQuasistatic::backwardDynamics () [protected]

Implements the Backward Dynamics step of the Featherstone algorithm.

Definition at line 67 of file ArticulatedAgentQuasistatic.cpp.

References ContactInfo::alpha, PhysicalObject::alpha, ArticulatedComponent::beta, ArticulatedComponent::betaExternal, ArticulatedComponent::betaStar, ArticulatedLink::childLinks, computeBodyDerivativesWithoutContacts(), PhysicalObject::contacts, ArticulatedLimb::cos2, ArticulatedLink::cos2, ArticulatedLimb::cosTheta, ArticulatedLink::cosTheta, ArticulatedComponent::deleteK(), PhysicalObject::externalForce, PhysicalObject::externalTorque, SymmetricMatrix3::getDeterminant(), Matrix3::getElement(), ArticulatedComponent::IStar, ArticulatedComponent::IStar0, ArticulatedComponent::K, ArticulatedLimb::l, ArticulatedLink::l, ArticulatedLimb::links, SymmetricMatrix3::mirror(), ArticulatedLink::motorTorque, ContactInfo::n, ContactInfo::pxn, real, Vector2::rotate(), SymmetricMatrix3::setElements(), Vector2::setToZero(), Vector3::setXYZ(), ContactInfo::sigma, ArticulatedLimb::sin2, ArticulatedLink::sin2, ArticulatedLimb::sinCos, ArticulatedLink::sinCos, ArticulatedLimb::sinTheta, ArticulatedLink::sinTheta, ArticulatedAgentBase::solveSystem(), sqr(), ArticulatedLink::torqueContact, PhysicalObject::v, Vector3::x, Vector2::x, Vector3::y, Vector2::y, and Vector3::z.

Referenced by computeDerivativesWithoutContacts().

4.2.3.2 void ArticulatedAgentQuasistatic::computeBodyDerivatives (GlobalContactInfoVector * *globalContacts*) [protected, virtual]

Computes the body derivatives. It is externalized in order to allow the possibility of a fixed body.

Definition at line 529 of file ArticulatedAgentQuasistatic.cpp.

References ArticulatedLink::force, real, Vector2::rotate(), Vector2::setXY(), and Vector2::x.

Referenced by computeDerivatives().

4.2.3.3 void ArticulatedAgentQuasistatic::computeBodyDerivativesWithoutContacts (real *determinant*) [protected, virtual]

Computes the body derivatives without contacts. It is externalized in order to allow the possibility of a fixed body.

Definition at line 363 of file ArticulatedAgentQuasistatic.cpp.

References `Vector2::rotate()`, `Vector2::setXY()`, `ArticulatedAgentBase::solveSystem()`, `Vector3::x`, `Vector3::y`, and `Vector3::z`.

Referenced by `backwardDynamics()`.

4.2.3.4 `void ArticulatedAgentQuasistatic::computeDerivatives (GlobalContactInfoVector * globalContacts)` [protected, virtual]

Computes the derivatives, given the contact forces. The `globalContacts` pointer is needed for contact processing for articulated agents.

Reimplemented from [PhysicalObject](#).

Definition at line 495 of file ArticulatedAgentQuasistatic.cpp.

References `computeBodyDerivatives()`, `ArticulatedLink::force`, `ArticulatedLimb::links`, `ArticulatedComponent::Q`, `real`, `ComposedPhysicalObject::setSensors()`, and `ArticulatedLink::thetaD`.

4.2.3.5 `virtual void ArticulatedAgentQuasistatic::computeDerivativesWithoutContacts (ContactSolver * contactSolver)` [inline, protected, virtual]

Computes the derivatives through the Featherstone algorithm.

Reimplemented from [PhysicalObject](#).

Definition at line 33 of file ArticulatedAgentQuasistatic.h.

References `backwardDynamics()`, and `forwardAccelerations()`.

4.2.3.6 `void ArticulatedAgentQuasistatic::computeForces (GlobalContactInfoVector * globalContacts)` [protected]

Definition at line 543 of file ArticulatedAgentQuasistatic.cpp.

References `ArticulatedLink::computeForceQuasistatic()`, and `ArticulatedLimb::links`.

4.2.3.7 `void ArticulatedAgentQuasistatic::computeTotalForces (GlobalContactInfoVector * globalContacts)` [protected]

Computes the total forces and torques acting on a link; used for testing purposes.

Definition at line 558 of file ArticulatedAgentQuasistatic.cpp.

References `ArticulatedLink::childLinks`, `ArticulatedLink::computeForceQuasistatic()`, `ArticulatedLink::cosTheta`, `ArticulatedLink::force`, `ArticulatedLink::forceLocal`, `ArticulatedLink::l`, `ArticulatedLimb::links`, `ArticulatedLink::motorTorque`, `ArticulatedLink::sinTheta`, `ArticulatedLink::totalForce`, `ArticulatedLink::totalTorque`, `Vector2::x`, and `Vector2::y`.

4.2.3.8 `void ArticulatedAgentQuasistatic::forwardAccelerations (ContactSolver * contactSolver)` [protected]

Implements the Forward Accelerations step of the Featherstone algorithm.

Definition at line 372 of file ArticulatedAgentQuasistatic.cpp.

References `ContactInfo::alpha`, `PhysicalObject::alpha`, `ArticulatedComponent::betaStar`, `ContactSolver::contactMatrix`, `PhysicalObject::contacts`, `ContactSolver::contactVector`, `ArticulatedLink::cosTheta`, `Vector2::cross()`, `Matrix3::getElement()`, `ArticulatedComponent::IStar`, `ArticulatedComponent::K`, `ArticulatedLink::l`, `ArticulatedLimb::l`, `ArticulatedLimb::links`, `ArticulatedLink::motorTorque`, `ContactInfo::n`, `ArticulatedComponent::nContacts`, `PhysicalObject::omega`, `ContactInfo::p`, `ArticulatedLink::parentLink`, `ArticulatedComponent::Q`, `real`, `Vector2::rotate()`, `Vector2::setXY()`, `Vector3::setXYZ()`, `ContactInfo::sigma`, `ArticulatedLink::sinTheta`, `ArticulatedLimb::theta`, `ArticulatedLink::thetaD`, `ArticulatedLink::torqueContact`, `PhysicalObject::v`, `ArticulatedComponent::vLocal`, `Vector3::x`, `Vector2::x`, `Vector3::y`, `Vector2::y`, and `Vector3::z`.

Referenced by `computeDerivativesWithoutContacts()`.

4.2.3.9 void ArticulatedAgentQuasistatic::forwardKinematics () [protected]

Based on the position of the body and the thetas, computes the position and angle of each of the links.

Definition at line 27 of file `ArticulatedAgentQuasistatic.cpp`.

References `PhysicalObject::alpha`, `ComposedPhysicalObject::computeMemberPositions()`, `ArticulatedLink::computeMotorTorque()`, `ArticulatedLink::computeSinCos()`, `ArticulatedLink::l`, `ArticulatedLimb::l`, `ArticulatedLimb::links`, `ArticulatedLink::parentLink`, `PhysicalObject::r`, `real`, `Vector2::rotate()`, `Vector2::setXY()`, `ArticulatedLink::theta`, and `ArticulatedLimb::theta`.

Referenced by `Iunctus::build()`, and `integrate()`.

4.2.3.10 void ArticulatedAgentQuasistatic::integrate (const Integrator & integrator) [protected, virtual]

Advances the time to the next timestep, updating the positions.

Reimplemented from `ComposedPhysicalObject`.

Definition at line 615 of file `ArticulatedAgentQuasistatic.cpp`.

References `PhysicalObject::boxMax`, `PhysicalObject::boxMin`, `forwardKinematics()`, `ComposedPhysicalObject::integrate()`, `ArticulatedLink::integrate()`, `ArticulatedLimb::links`, `Vector2::updateMax()`, and `Vector2::updateMin()`.

4.2.3.11 void ArticulatedAgentQuasistatic::rollback () [protected, virtual]

Rolls back the time to the previous timestep.

Reimplemented from `ComposedPhysicalObject`.

Definition at line 643 of file `ArticulatedAgentQuasistatic.cpp`.

References `PhysicalObject::boxMax`, `PhysicalObject::boxMin`, `ArticulatedLimb::links`, `ComposedPhysicalObject::rollback()`, `ArticulatedLink::rollback()`, `Vector2::updateMax()`, and `Vector2::updateMin()`.

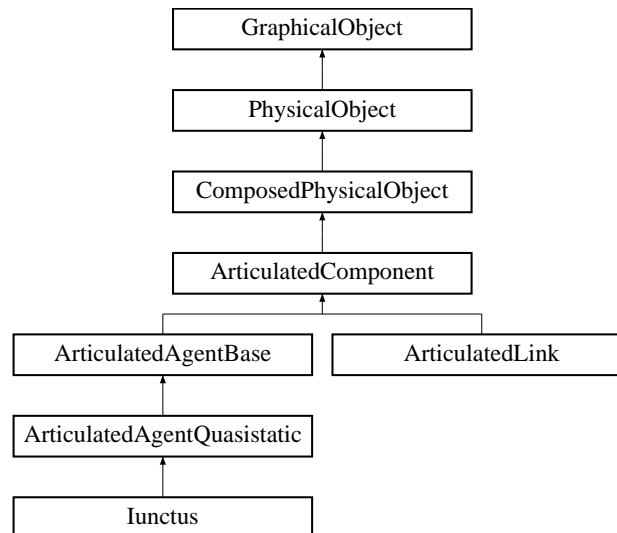
The documentation for this class was generated from the following files:

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

4.3 ArticulatedComponent Class Reference

```
#include <ArticulatedComponent.h>
```

Inheritance diagram for ArticulatedComponent::



Public Member Functions

- [ArticulatedComponent](#) (std::string [label](#))
- virtual [~ArticulatedComponent](#) ()
- void [computeIStar0](#) ()
- void [deleteK](#) ()

Public Attributes

- [SymmetricMatrix3](#) [IStar](#)
- [SymmetricMatrix3](#) [IStar0](#)
- [Vector3](#) [beta](#)
- [Vector3](#) [betaExternal](#)
- [Vector3](#) [betaStar](#)
- [Vector2](#) [aLocal](#)
- [Vector2](#) [vLocal](#)
- unsigned int [nContacts](#)
- [LinkContactInfoPVector](#) [K](#)
- [LinkContactInfoVector](#) [Q](#)

4.3.1 Detailed Description

A component for articulated agents: it may represent the body of the agent, or (extended) an articulated link.

Definition at line 16 of file `ArticulatedComponent.h`.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 ArticulatedComponent::ArticulatedComponent (std::string *label*)

Definition at line 12 of file ArticulatedComponent.cpp.

4.3.2.2 ArticulatedComponent::~~ArticulatedComponent () [virtual]

Destructor; deletes K.

Definition at line 17 of file ArticulatedComponent.cpp.

References deleteK().

4.3.3 Member Function Documentation

4.3.3.1 void ArticulatedComponent::computeIStar0 ()

Computes the value of IStar0, based on the mass and moment of inertia of the object.

Definition at line 21 of file ArticulatedComponent.cpp.

References IStar, IStar0, real, Matrix3::setRow(), Vector2::x, and Vector2::y.

Referenced by ArticulatedLimb::addOneObjectLink(), and Iunctus::build().

4.3.3.2 void ArticulatedComponent::deleteK ()

Deletes K and its components.

Definition at line 31 of file ArticulatedComponent.cpp.

References K, and purgeContainer().

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), and ~ArticulatedComponent().

4.3.4 Member Data Documentation

4.3.4.1 Vector2 ArticulatedComponent::aLocal

The acceleration of the object relative to the LRS, but expressed in the local (object) reference system (ORS); used in the classical model

Definition at line 32 of file ArticulatedComponent.h.

4.3.4.2 Vector3 ArticulatedComponent::beta

Definition at line 26 of file ArticulatedComponent.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics().

4.3.4.3 Vector3 ArticulatedComponent::betaExternal

Definition at line 27 of file ArticulatedComponent.h.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`.

4.3.4.4 **Vector3 ArticulatedComponent::betaStar**

Definition at line 28 of file `ArticulatedComponent.h`.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, and `ArticulatedAgentQuasistatic::forwardAccelerations()`.

4.3.4.5 **SymmetricMatrix3 ArticulatedComponent::IStar**

Parameters used by the Featherstone algorithm:

Definition at line 24 of file `ArticulatedComponent.h`.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, `computeIStar0()`, and `ArticulatedAgentQuasistatic::forwardAccelerations()`.

4.3.4.6 **SymmetricMatrix3 ArticulatedComponent::IStar0**

Definition at line 25 of file `ArticulatedComponent.h`.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, and `computeIStar0()`.

4.3.4.7 **LinkContactInfoPVector ArticulatedComponent::K**

Link contact information used by the Featherstone algorithm.

Definition at line 44 of file `ArticulatedComponent.h`.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, `deleteK()`, and `ArticulatedAgentQuasistatic::forwardAccelerations()`.

4.3.4.8 **unsigned int ArticulatedComponent::nContacts**

Definition at line 41 of file `ArticulatedComponent.h`.

Referenced by `ArticulatedAgentQuasistatic::forwardAccelerations()`.

4.3.4.9 **LinkContactInfoVector ArticulatedComponent::Q**

Link contact information used by the Featherstone algorithm. This list contains objects, not pointers.

Definition at line 48 of file `ArticulatedComponent.h`.

Referenced by `ArticulatedAgentQuasistatic::computeDerivatives()`, and `ArticulatedAgentQuasistatic::forwardAccelerations()`.

4.3.4.10 **Vector2 ArticulatedComponent::vLocal**

The velocity of the object relative to the LRS, but expressed in the local (object) reference system (ORS); used in the quasistatic model

Definition at line 36 of file `ArticulatedComponent.h`.

Referenced by `ArticulatedAgentQuasistatic::forwardAccelerations()`.

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

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

4.4 ArticulatedLimb Class Reference

```
#include <ArticulatedLimb.h>
```

Public Member Functions

- [ArticulatedLimb](#) ([real l](#), [real theta](#))
- virtual [~ArticulatedLimb](#) ()
- void [computeSinCos](#) ()
- void [setProperties](#) ([real l](#), [real theta](#))
- [ArticulatedLink](#) * [addLink](#) ()
- [ArticulatedLink](#) * [addOneObjectLink](#) ([real l](#), [real theta0](#), [PhysicalObject](#) *object, [real x](#), [real y](#), [real alpha](#), [ArticulatedLink](#) *parent=NULL, bool hasElasticTorque=true)
- virtual void [draw](#) ([GUI](#) *gui)
- virtual void [setOutlineColor](#) ([Color](#) color)
- virtual void [setFillColor](#) ([Color](#) color)
- void [deleteLinks](#) ()

Public Attributes

- std::string [label](#)
- [real l](#)
- [real theta](#)
- [real sinTheta](#)
- [real cosTheta](#)
- [real sin2](#)
- [real cos2](#)
- [real sinCos](#)
- [ArticulatedLinkPVector](#) [links](#)

4.4.1 Constructor & Destructor Documentation

4.4.1.1 [ArticulatedLimb::ArticulatedLimb](#) ([real l](#), [real theta](#))

Constructor, sets l and theta.

Definition at line 12 of file [ArticulatedLimb.cpp](#).

4.4.1.2 [ArticulatedLimb::~~ArticulatedLimb](#) () [virtual]

Destructor, deletes the links.

Definition at line 18 of file [ArticulatedLimb.cpp](#).

4.4.2 Member Function Documentation

4.4.2.1 [ArticulatedLink](#) * ArticulatedLimb::addLink ()

Definition at line 22 of file ArticulatedLimb.cpp.

References label, and links.

Referenced by addOneObjectLink().

4.4.2.2 [ArticulatedLink](#) * ArticulatedLimb::addOneObjectLink (real l, real theta0, [PhysicalObject](#) * object, real x, real y, real alpha, [ArticulatedLink](#) * parent = NULL, bool hasElasticTorque = true)

Adds a link composed of one object. The link will be articulated around a point situated at distance l on the parent link.

Definition at line 31 of file ArticulatedLimb.cpp.

References addLink(), ComposedPhysicalObject::addObject(), ArticulatedComponent::computeIStar0(), ArticulatedLink::computeSinCos(), ArticulatedLink::hasElasticTorque, ArticulatedLink::l, ArticulatedLink::setParentLink(), ArticulatedLink::theta, and ArticulatedLink::theta0.

Referenced by Iunctus::build().

4.4.2.3 void ArticulatedLimb::computeSinCos ()

Updates the values of sin, cos, sin2, cos2, sinCos as a function of theta

Definition at line 48 of file ArticulatedLimb.cpp.

References cos2, cosTheta, sin2, sinCos, sinTheta, and theta.

Referenced by setProperties().

4.4.2.4 void ArticulatedLimb::deleteLinks ()

Deletes the links list and the member links.

Definition at line 62 of file ArticulatedLimb.cpp.

References links, and purgeContainer().

4.4.2.5 void ArticulatedLimb::draw ([GUI](#) * gui) [virtual]

Definition at line 66 of file ArticulatedLimb.cpp.

References ArticulatedLink::draw(), and links.

4.4.2.6 void ArticulatedLimb::setFillColor ([Color](#) color) [virtual]

Definition at line 83 of file ArticulatedLimb.cpp.

References links, and ComposedPhysicalObject::setFillColor().

4.4.2.7 void ArticulatedLimb::setOutlineColor (Color color) [virtual]

Definition at line 76 of file ArticulatedLimb.cpp.

References links, and ComposedPhysicalObject::setOutlineColor().

4.4.2.8 void ArticulatedLimb::setProperties (real l, real theta)

Sets l, theta, calls computeSinCos().

Definition at line 56 of file ArticulatedLimb.cpp.

References computeSinCos().

Referenced by ArticulatedAgentBase::addLimb().

4.4.3 Member Data Documentation

4.4.3.1 real ArticulatedLimb::cos2

Functions of theta

Definition at line 46 of file ArticulatedLimb.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), and computeSinCos().

4.4.3.2 real ArticulatedLimb::cosTheta

Functions of theta

Definition at line 46 of file ArticulatedLimb.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), and computeSinCos().

4.4.3.3 real ArticulatedLimb::l

Definition at line 42 of file ArticulatedLimb.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), ArticulatedAgentQuasistatic::forwardAccelerations(), and ArticulatedAgentQuasistatic::forwardKinematics().

4.4.3.4 std::string ArticulatedLimb::label

Definition at line 39 of file ArticulatedLimb.h.

Referenced by ArticulatedAgentBase::addLimb(), and addLink().

4.4.3.5 ArticulatedLinkPVector ArticulatedLimb::links

A list containing the links that compose the limb. The list owns the links.

Definition at line 50 of file ArticulatedLimb.h.

Referenced by addLink(), ArticulatedAgentQuasistatic::backwardDynamics(), ArticulatedAgentQuasistatic::computeDerivatives(), ArticulatedAgentQuasistatic::computeForces(), Articulated-

AgentQuasistatic::computeTotalForces(), Iunctus::controll(), ArticulatedAgentBase::deleteContacts(), deleteLinks(), ArticulatedAgentBase::detectContacts(), ArticulatedAgentBase::detectInternalContacts(), ArticulatedAgentBase::detectMouseContact(), draw(), ArticulatedAgentQuasistatic::forwardAccelerations(), ArticulatedAgentQuasistatic::forwardKinematics(), ArticulatedAgentQuasistatic::integrate(), Iunctus::proprioception(), ArticulatedAgentBase::registerPrimitives(), ArticulatedAgentQuasistatic::rollback(), setFillColor(), and setOutlineColor().

4.4.3.6 [real ArticulatedLimb::sin2](#)

Functions of theta

Definition at line 46 of file ArticulatedLimb.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), and computeSinCos().

4.4.3.7 [real ArticulatedLimb::sinCos](#)

Functions of theta

Definition at line 46 of file ArticulatedLimb.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), and computeSinCos().

4.4.3.8 [real ArticulatedLimb::sinTheta](#)

Functions of theta

Definition at line 46 of file ArticulatedLimb.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), and computeSinCos().

4.4.3.9 [real ArticulatedLimb::theta](#)

Definition at line 43 of file ArticulatedLimb.h.

Referenced by computeSinCos(), ArticulatedAgentQuasistatic::forwardAccelerations(), and ArticulatedAgentQuasistatic::forwardKinematics().

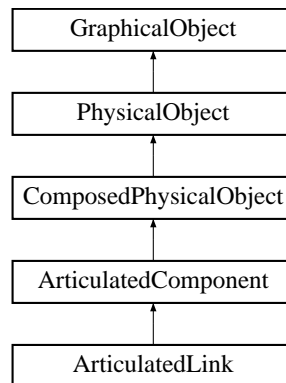
The documentation for this class was generated from the following files:

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

4.5 ArticulatedLink Class Reference

```
#include <ArticulatedLink.h>
```

Inheritance diagram for ArticulatedLink::



Public Member Functions

- [ArticulatedLink](#) (const std::string &label)
- virtual [~ArticulatedLink](#) ()
- void [computeForceQuasistatic](#) (GlobalContactInfoVector *globalContacts)
- void [normalizeTheta](#) ()
- void [computeSinCos](#) ()
- void [integrate](#) (const Integrator &integrator)
- void [rollback](#) ()
- void [computeMotorTorque](#) ()
- void [setParentLink](#) (ArticulatedLink *parentLink)
- void [fillContactMatrix](#) (ContactSolver *contactSolver, ContactInfo *contactInfo)
- void [deleteContacts](#) ()
- void [detectTorqueContact](#) (GlobalContactInfoVector *contacts)
- void [draw](#) (GUI *gui)

Public Attributes

- bool [hasElasticTorque](#)
- ContactInfo * [torqueContact](#)
- real [totalTorque](#)
- Vector2 [force](#)
- Vector2 [forceLocal](#)
- Vector2 [totalForce](#)
- real [theta](#)
- real [thetaOld](#)
- real [sinTheta](#)
- real [cosTheta](#)
- real [sin2](#)
- real [cos2](#)
- real [sinCos](#)

- [Vector3 coriolis](#)
- [real thetaD](#)
- [real thetaDD](#)
- [real theta0](#)
- [real theta0Min](#)
- [real theta0Max](#)
- [real deltaThetaMax](#)
- [real motorTorque](#)
- [real k](#)
- [real eta](#)
- [real l](#)
- [ArticulatedLink * parentLink](#)
- [ArticulatedLinkPVector childLinks](#)

4.5.1 Detailed Description

A link that composes an articulated agent. A link has an articulation on which the agent can produce a torque.

Definition at line 24 of file ArticulatedLink.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 ArticulatedLink::ArticulatedLink (const std::string & label)

Definition at line 14 of file ArticulatedLink.cpp.

References [deltaThetaMax](#), [eta](#), [k](#), [motorTorque](#), [parentLink](#), [theta](#), [theta0](#), [theta0Max](#), [theta0Min](#), [thetaD](#), [thetaDD](#), and [torqueContact](#).

4.5.2.2 ArticulatedLink::~~ArticulatedLink () [virtual]

Definition at line 30 of file ArticulatedLink.cpp.

4.5.3 Member Function Documentation

4.5.3.1 void ArticulatedLink::computeForceQuasistatic ([GlobalContactInfoVector](#) * globalContacts)

Computes the torque and force that acts on the link (from IStar and betaStar), for the quasistatical physics case.

Definition at line 69 of file ArticulatedLink.cpp.

References [force](#), [Matrix3::getElement\(\)](#), [Vector2::rotate\(\)](#), [Vector2::setToZero\(\)](#), [Vector3::x](#), [Vector2::x](#), [Vector2::y](#), [Vector3::y](#), and [Vector3::z](#).

Referenced by [ArticulatedAgentQuasistatic::computeForces\(\)](#), and [ArticulatedAgentQuasistatic::computeTotalForces\(\)](#).

4.5.3.2 void ArticulatedLink::computeMotorTorque ()

Computes the motor torque, given theta and (target theta) theta0, as an elastic torque.

Definition at line 45 of file ArticulatedLink.cpp.

References k, motorTorque, and theta.

Referenced by ArticulatedAgentQuasistatic::forwardKinematics().

4.5.3.3 void ArticulatedLink::computeSinCos ()

Updates the values of sin, cos, sin2, cos2, sinCos as a function of theta

Definition at line 53 of file ArticulatedLink.cpp.

References cos2, cosTheta, sin2, sinCos, sinTheta, and theta.

Referenced by ArticulatedLimb::addOneObjectLink(), and ArticulatedAgentQuasistatic::forwardKinematics().

4.5.3.4 void ArticulatedLink::deleteContacts () [inline, virtual]

Delete the contacts of the object.

Reimplemented from [PhysicalObject](#).

Definition at line 65 of file ArticulatedLink.h.

References PhysicalObject::deleteContacts(), force, Vector2::setToZero(), and torqueContact.

4.5.3.5 void ArticulatedLink::detectTorqueContact ([GlobalContactInfoVector](#) * contacts)

Definition at line 98 of file ArticulatedLink.cpp.

References real, theta, theta0Max, and torqueContact.

4.5.3.6 void ArticulatedLink::draw ([GUI](#) * gui) [virtual]

Calls the draw method for the member objects.

Reimplemented from [ComposedPhysicalObject](#).

Definition at line 106 of file ArticulatedLink.cpp.

References ComposedPhysicalObject::draw(), GUI::drawCircle(), real, GUI::setPenColor(), Vector2::x, and Vector2::y.

Referenced by ArticulatedLimb::draw().

4.5.3.7 void ArticulatedLink::fillContactMatrix ([ContactSolver](#) * contactSolver, [ContactInfo](#) * contactInfo) [inline, virtual]

Does nothing. The contact matrix filling for links is performed in ArticulatedAgent::forwardAccelerations.

Reimplemented from [PhysicalObject](#).

Definition at line 61 of file ArticulatedLink.h.

4.5.3.8 void ArticulatedLink::integrate (const [Integrator](#) & *integrator*) [virtual]

Advances the time to the next timestep. A further call to forward kinematics is needed for updating the global coordinates.

Reimplemented from [ComposedPhysicalObject](#).

Definition at line 33 of file ArticulatedLink.cpp.

References [Integrator::integrate\(\)](#), [theta](#), [thetaD](#), and [thetaOld](#).

Referenced by [ArticulatedAgentQuasistatic::integrate\(\)](#).

4.5.3.9 void ArticulatedLink::normalizeTheta ()

Normalizes theta to the interval $(-\pi, \pi)$ around [theta0](#)

Definition at line 38 of file ArticulatedLink.cpp.

References [theta](#), and [theta0](#).

4.5.3.10 void ArticulatedLink::rollback () [inline, virtual]

Rolls back the time to the previous timestep.

Reimplemented from [ComposedPhysicalObject](#).

Definition at line 44 of file ArticulatedLink.h.

References [ComposedPhysicalObject::rollback\(\)](#), and [theta](#).

Referenced by [ArticulatedAgentQuasistatic::rollback\(\)](#).

4.5.3.11 void ArticulatedLink::setParentLink ([ArticulatedLink](#) * *parentLink*)

Definition at line 61 of file ArticulatedLink.cpp.

References [childLinks](#).

Referenced by [ArticulatedLimb::addOneObjectLink\(\)](#).

4.5.4 Member Data Documentation**4.5.4.1 [ArticulatedLinkPVector](#) ArticulatedLink::childLinks**

A list of child links. The list does NOT own the links. It should NOT delete the child links on destruction.

Definition at line 125 of file ArticulatedLink.h.

Referenced by [ArticulatedAgentQuasistatic::backwardDynamics\(\)](#), [ArticulatedAgentQuasistatic::computeTotalForces\(\)](#), and [setParentLink\(\)](#).

4.5.4.2 [Vector3](#) ArticulatedLink::coriolis

The Coriolis force.

Definition at line 96 of file ArticulatedLink.h.

4.5.4.3 **real** `ArticulatedLink::cos2`

Functions of theta

Definition at line 93 of file `ArticulatedLink.h`.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, and `computeSinCos()`.

4.5.4.4 **real** `ArticulatedLink::cosTheta`

Functions of theta

Definition at line 93 of file `ArticulatedLink.h`.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, `computeSinCos()`, `ArticulatedAgentQuasistatic::computeTotalForces()`, and `ArticulatedAgentQuasistatic::forwardAccelerations()`.

4.5.4.5 **real** `ArticulatedLink::deltaThetaMax`

The maximum permitted displacement relative to the equilibrium position of the link

Definition at line 108 of file `ArticulatedLink.h`.

Referenced by `ArticulatedLink()`.

4.5.4.6 **real** `ArticulatedLink::eta`

Damping factor of the link articulation

Definition at line 115 of file `ArticulatedLink.h`.

Referenced by `ArticulatedLink()`.

4.5.4.7 **Vector2** `ArticulatedLink::force`

The force that is exercised on the link by its parent link, in laboratory reference system.

Definition at line 81 of file `ArticulatedLink.h`.

Referenced by `ArticulatedAgentQuasistatic::computeBodyDerivatives()`, `ArticulatedAgentQuasistatic::computeDerivatives()`, `computeForceQuasistatic()`, `ArticulatedAgentQuasistatic::computeTotalForces()`, and `deleteContacts()`.

4.5.4.8 **Vector2** `ArticulatedLink::forceLocal`

The force that is exercised on the link by its parent link, in link reference system.

Definition at line 83 of file `ArticulatedLink.h`.

Referenced by `ArticulatedAgentQuasistatic::computeTotalForces()`.

4.5.4.9 **bool** `ArticulatedLink::hasElasticTorque`

Determines whether the torque should be computed as an elastic torque given by theta and theta0 (if true) or whether the torque is given at each timestep (if false)

Definition at line 52 of file ArticulatedLink.h.

Referenced by ArticulatedLimb::addOneObjectLink().

4.5.4.10 **real ArticulatedLink::k**

Elastic constant of the link articulation

Definition at line 113 of file ArticulatedLink.h.

Referenced by ArticulatedLink(), Iunctus::build(), and computeMotorTorque().

4.5.4.11 **real ArticulatedLink::l**

Length of the link along its x axis

Definition at line 118 of file ArticulatedLink.h.

Referenced by ArticulatedLimb::addOneObjectLink(), ArticulatedAgentQuasistatic::backwardDynamics(), ArticulatedAgentQuasistatic::computeTotalForces(), ArticulatedAgentQuasistatic::forwardAccelerations(), and ArticulatedAgentQuasistatic::forwardKinematics().

4.5.4.12 **real ArticulatedLink::motorTorque**

The motor torque applied at the link joint

Definition at line 110 of file ArticulatedLink.h.

Referenced by ArticulatedLink(), ArticulatedAgentQuasistatic::backwardDynamics(), computeMotorTorque(), ArticulatedAgentQuasistatic::computeTotalForces(), and ArticulatedAgentQuasistatic::forwardAccelerations().

4.5.4.13 **ArticulatedLink* ArticulatedLink::parentLink**

The parent link.

Definition at line 121 of file ArticulatedLink.h.

Referenced by ArticulatedLink(), ArticulatedAgentQuasistatic::forwardAccelerations(), and ArticulatedAgentQuasistatic::forwardKinematics().

4.5.4.14 **real ArticulatedLink::sin2**

Functions of theta

Definition at line 93 of file ArticulatedLink.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), and computeSinCos().

4.5.4.15 **real ArticulatedLink::sinCos**

Functions of theta

Definition at line 93 of file ArticulatedLink.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), and computeSinCos().

4.5.4.16 **real ArticulatedLink::sinTheta**

Functions of theta

Definition at line 93 of file ArticulatedLink.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), computeSinCos(), ArticulatedAgentQuasistatic::computeTotalForces(), and ArticulatedAgentQuasistatic::forwardAccelerations().

4.5.4.17 **real ArticulatedLink::theta**

The rotation angle of the link relative to the coordinate system of the predecessor link. It is the generalized coordinate of the link.

Definition at line 89 of file ArticulatedLink.h.

Referenced by ArticulatedLimb::addOneObjectLink(), ArticulatedLink(), computeMotorTorque(), computeSinCos(), detectTorqueContact(), ArticulatedAgentQuasistatic::forwardKinematics(), integrate(), normalizeTheta(), Iunctus::proprioception(), and rollback().

4.5.4.18 **real ArticulatedLink::theta0**

The equilibrium position of the link, commanded by the motor neurons

Definition at line 104 of file ArticulatedLink.h.

Referenced by ArticulatedLimb::addOneObjectLink(), ArticulatedLink(), Iunctus::build(), Iunctus::controll(), and normalizeTheta().

4.5.4.19 **real ArticulatedLink::theta0Max**

Minimum and maximum values of theta0

Definition at line 106 of file ArticulatedLink.h.

Referenced by ArticulatedLink(), Iunctus::build(), Iunctus::controll(), detectTorqueContact(), and Iunctus::proprioception().

4.5.4.20 **real ArticulatedLink::theta0Min**

Minimum and maximum values of theta0

Definition at line 106 of file ArticulatedLink.h.

Referenced by ArticulatedLink(), Iunctus::build(), Iunctus::controll(), and Iunctus::proprioception().

4.5.4.21 **real ArticulatedLink::thetaD**

Temporal derivate of theta

Definition at line 99 of file ArticulatedLink.h.

Referenced by ArticulatedLink(), ArticulatedAgentQuasistatic::computeDerivatives(), ArticulatedAgentQuasistatic::forwardAccelerations(), and integrate().

4.5.4.22 [real ArticulatedLink::thetaDD](#)

Definition at line 101 of file ArticulatedLink.h.

Referenced by ArticulatedLink().

4.5.4.23 [real ArticulatedLink::thetaOld](#)

Definition at line 90 of file ArticulatedLink.h.

Referenced by integrate().

4.5.4.24 [ContactInfo* ArticulatedLink::torqueContact](#)

Definition at line 64 of file ArticulatedLink.h.

Referenced by ArticulatedLink(), ArticulatedAgentQuasistatic::backwardDynamics(), deleteContacts(), detectTorqueContact(), and ArticulatedAgentQuasistatic::forwardAccelerations().

4.5.4.25 [Vector2 ArticulatedLink::totalForce](#)

The total force that is exercised on the link, in laboratory reference system.

Definition at line 85 of file ArticulatedLink.h.

Referenced by ArticulatedAgentQuasistatic::computeTotalForces().

4.5.4.26 [real ArticulatedLink::totalTorque](#)

The total torque that acts on the link, relative to its reference point.

Definition at line 79 of file ArticulatedLink.h.

Referenced by ArticulatedAgentQuasistatic::computeTotalForces().

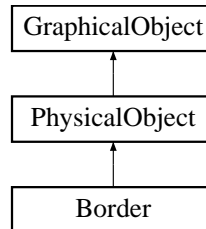
The documentation for this class was generated from the following files:

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

4.6 Border Class Reference

```
#include <Border.h>
```

Inheritance diagram for Border::



Public Member Functions

- **Border** (**real** x, **real** y, **real** normalX, **real** normalY, std::string label="")
- virtual ~**Border** ()
- virtual bool **detectContacts** (**PhysicalObject** *object, **GlobalContactInfoVector** *contacts)
- virtual bool **detectContacts** (**Border** *object, **GlobalContactInfoVector** *contacts)
- virtual void **computeDerivativesWithoutContacts** (**ContactSolver** *contactSolver)
- virtual void **integrate** (const **Integrator** &integrator)
- virtual void **rollback** ()
- void **fillContactMatrix** (**ContactSolver** *contactSolver, **ContactInfo** *contact)
- void **draw** (**GUI** *gui)

Public Attributes

- **Vector2** normal

4.6.1 Detailed Description

A border object: a line that delimitates a semiplane in the simulation space. The line has infinite length; it is defined by one point belonging to the line and the normal to this line, oriented towards the interior (the zone where the simulated objects evolve). The location of the reference point r on the line does not matter.

Definition at line 17 of file Border.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 **Border::Border** (**real** x, **real** y, **real** normalX, **real** normalY, std::string label = " ")

Constructor, sets the position and the normal.

Definition at line 12 of file Border.cpp.

References normal, Vector2::x, and Vector2::y.

4.6.2.2 `Border::~~Border()` [virtual]

Destructor, does nothing.

Definition at line 49 of file `Border.cpp`.

4.6.3 Member Function Documentation

4.6.3.1 `virtual void Border::computeDerivativesWithoutContacts(ContactSolver * contactSolver)` [inline, virtual]

Does nothing (the border is fixed). Overrides method defined in [PhysicalObject](#).

Reimplemented from [PhysicalObject](#).

Definition at line 32 of file `Border.h`.

4.6.3.2 `virtual bool Border::detectContacts(Border * object, GlobalContactInfoVector * contacts)` [inline, virtual]

No contact is possible between two border objects.

Reimplemented from [PhysicalObject](#).

Definition at line 29 of file `Border.h`.

4.6.3.3 `virtual bool Border::detectContacts(PhysicalObject * object, GlobalContactInfoVector * contacts)` [inline, virtual]

Contact handling.

Implements [PhysicalObject](#).

Definition at line 24 of file `Border.h`.

References `PhysicalObject::detectContacts()`.

4.6.3.4 `void Border::draw(GUI * gui)` [inline, virtual]

Draw the object to the graphical user interface.

Implements [GraphicalObject](#).

Definition at line 38 of file `Border.h`.

4.6.3.5 `void Border::fillContactMatrix(ContactSolver * contactSolver, ContactInfo * contact)` [inline, virtual]

Computes the contribution of the current object to the matrix of interactions between all objects that are in contact.

Reimplemented from [PhysicalObject](#).

Definition at line 37 of file `Border.h`.

4.6.3.6 virtual void Border::integrate (const [Integrator](#) & *integrator*) [`inline`, `virtual`]

Advances the time to the next timestep.

Reimplemented from [PhysicalObject](#).

Definition at line 34 of file Border.h.

4.6.3.7 virtual void Border::rollback () [`inline`, `virtual`]

Rolls back the time to the previous timestep.

Reimplemented from [PhysicalObject](#).

Definition at line 35 of file Border.h.

4.6.4 Member Data Documentation**4.6.4.1 [Vector2](#) Border::normal**

The normal to the border.

Definition at line 41 of file Border.h.

Referenced by [Border\(\)](#), [Circle::detectContacts\(\)](#), and [CappedRectangle::detectContacts\(\)](#).

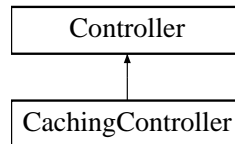
The documentation for this class was generated from the following files:

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

4.7 CachingController Class Reference

```
#include <CachingController.h>
```

Inheritance diagram for CachingController::



Public Member Functions

- [CachingController](#) (unsigned int [nInputs](#), unsigned int [nOutputs](#))
- virtual [~CachingController](#) ()

Protected Member Functions

- void [cacheInput](#) ()

Protected Attributes

- float * [oldInput](#)

4.7.1 Detailed Description

An abstract controller that caches its input. This may be useful when the output(t) depends on both input(t) and input (t-dt).

Definition at line 10 of file CachingController.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 CachingController::CachingController (unsigned int *nInputs*, unsigned int *nOutputs*)

Definition at line 4 of file CachingController.cpp.

References [oldInput](#).

4.7.2.2 CachingController::~~CachingController () [virtual]

Definition at line 12 of file CachingController.cpp.

4.7.3 Member Function Documentation

4.7.3.1 void CachingController::cacheInput () [inline, protected]

Copies input to [oldInput](#).

Definition at line 19 of file CachingController.h.

References `oldInput`.

4.7.4 Member Data Documentation

4.7.4.1 `float* CachingController::oldInput` [protected]

Definition at line 16 of file CachingController.h.

Referenced by `cacheInput()`, and `CachingController()`.

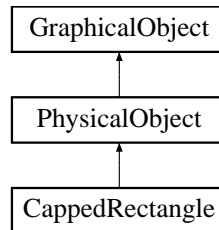
The documentation for this class was generated from the following files:

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

4.8 CappedRectangle Class Reference

```
#include <CappedRectangle.h>
```

Inheritance diagram for CappedRectangle::



Public Member Functions

- [CappedRectangle](#) ([real l](#), [real R](#), [real x=0](#), [real y=0](#), [real alpha=0](#), [std::string label=""](#), [int nSensorsLateral=0](#), [int nSensorsSemicircle=0](#), [real saturationForce=0.5](#), [Color outlineColor=GUI::colorBlack](#), [Color fillColor=GUI::colorTransparent](#))
- virtual [~CappedRectangle](#) ()
- virtual void [computeMass](#) ([real density=ThyrixParameters::defaultDensity](#))
- virtual void [computeInertia](#) ()
- virtual bool [detectContacts](#) ([PhysicalObject *object](#), [GlobalContactInfoVector *contacts](#))
- bool [detectContacts](#) ([Border *border](#), [GlobalContactInfoVector *contacts](#))
- bool [detectContacts](#) ([Circle *circle](#), [GlobalContactInfoVector *contacts](#))
- bool [detectContacts](#) ([CappedRectangle *capsule](#), [GlobalContactInfoVector *contacts](#))
- bool [detectMouseContact](#) ([const Vector2 &rMouse](#), [Vector2 &p](#), [PhysicalObject *&object](#))
- virtual void [computeBox](#) ()
- virtual void [setSensor](#) ([ContactInfo *contact](#))
- void [draw](#) ([GUI *gui](#))
- void [drawSensors](#) ([GUI *gui](#))

Public Attributes

- [real l](#)
- [real R](#)
- [real cosAlpha](#)
- [real sinAlpha](#)
- unsigned [nSensorsLateral](#)
- unsigned [nSensorsSemicircle](#)

Private Member Functions

- int [getISensorSemicircle](#) ([Vector2 &p](#), [int sign](#))
- void [setISensorLateral](#) ([int &i1](#), [int &i2](#), [Vector2 &p](#), [real e1](#), [real e2](#))
- int [getISensorLateral](#) ([Vector2 &p](#), [real e](#))
- int [getISensorLateral](#) ([real side](#), [real e](#))
- [real getLateralSide](#) ([Vector2 &p](#))

4.8.1 Detailed Description

Defines a rectangle having the thin ends capped with half circles. The base position (relative to which the rotation angle is measured) is along the x axis, like this: (_____). l is the half length of the rectangle, R its half width. The end half circles have a radius R .

Definition at line 13 of file CappedRectangle.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 CappedRectangle::CappedRectangle (**real** l , **real** R , **real** $x = 0$, **real** $y = 0$, **real** $\alpha = 0$, **std::string** $label = ""$, **int** $nSensorsLateral = 0$, **int** $nSensorsSemicircle = 0$, **real** $saturationForce = 0.5$, **Color** $outlineColor = GUI::colorBlack$, **Color** $fillColor = GUI::colorTransparent$)

Definition at line 6 of file CappedRectangle.cpp.

References `computeBox()`, `computeInertia()`, `computeMass()`, `Vector2::x`, and `Vector2::y`.

4.8.2.2 CappedRectangle::~CappedRectangle () [virtual]

Destructor, does nothing.

Definition at line 26 of file CappedRectangle.cpp.

4.8.3 Member Function Documentation

4.8.3.1 virtual void CappedRectangle::computeBox () [inline, virtual]

Computes the coordinates of the bounding box. It is called after each call to `integrate()` or `rollback()`.

Reimplemented from `PhysicalObject`.

Definition at line 50 of file CappedRectangle.h.

References `cosAlpha`, and `sinAlpha`.

Referenced by `CappedRectangle()`.

4.8.3.2 void CappedRectangle::computeInertia () [virtual]

Computes the moment of inertia around the z axis that passes through the center

Reimplemented from `PhysicalObject`.

Definition at line 34 of file CappedRectangle.cpp.

References l , and R .

Referenced by `CappedRectangle()`.

4.8.3.3 void CappedRectangle::computeMass (**real** $density = ThyrixParameters::defaultDensity$) [virtual]

Sets the mass based on the dimensions and the density

Reimplemented from [PhysicalObject](#).

Definition at line 29 of file CappedRectangle.cpp.

References [l](#), and [R](#).

Referenced by [CappedRectangle\(\)](#).

4.8.3.4 **bool CappedRectangle::detectContacts** ([CappedRectangle](#) * *capsule*, [GlobalContactInfoVector](#) * *contacts*) [[virtual](#)]

Tests for contacts between this object and another capped rectangle

Reimplemented from [PhysicalObject](#).

Definition at line 299 of file CappedRectangle.cpp.

References [cosAlpha](#), [getISensorLateral\(\)](#), [getISensorSemicircle\(\)](#), [Vector2::getModule\(\)](#), [Vector2::getSquaredModule\(\)](#), [l](#), [M_PI](#), [Vector2::normalize\(\)](#), [PhysicalObject::r](#), [R](#), [real](#), [setISensorLateral\(\)](#), [Vector2::setXY\(\)](#), [sinAlpha](#), [sqr\(\)](#), [Vector2::x](#), and [Vector2::y](#).

4.8.3.5 **bool CappedRectangle::detectContacts** ([Circle](#) * *circle*, [GlobalContactInfoVector](#) * *contacts*) [[virtual](#)]

Tests for contacts between this object and a circle.

Reimplemented from [PhysicalObject](#).

Definition at line 189 of file CappedRectangle.cpp.

References [cosAlpha](#), [Circle::getISensor\(\)](#), [getISensorLateral\(\)](#), [getISensorSemicircle\(\)](#), [Vector2::getModule\(\)](#), [l](#), [M_PI](#), [Vector2::normalize\(\)](#), [Circle::R](#), [R](#), [PhysicalObject::r](#), [real](#), [Vector2::setXY\(\)](#), [sinAlpha](#), [sqr\(\)](#), [Vector2::x](#), and [Vector2::y](#).

4.8.3.6 **bool CappedRectangle::detectContacts** ([Border](#) * *border*, [GlobalContactInfoVector](#) * *contacts*) [[virtual](#)]

Tests for contacts between this object and a border.

Reimplemented from [PhysicalObject](#).

Definition at line 92 of file CappedRectangle.cpp.

References [cosAlpha](#), [getISensorSemicircle\(\)](#), [l](#), [Border::normal](#), [PhysicalObject::r](#), [real](#), [setISensorLateral\(\)](#), [Vector2::setToZero\(\)](#), [sinAlpha](#), [sqr\(\)](#), [Vector2::x](#), and [Vector2::y](#).

4.8.3.7 **virtual bool CappedRectangle::detectContacts** ([PhysicalObject](#) * *object*, [GlobalContactInfoVector](#) * *contacts*) [[inline](#), [virtual](#)]

Contact handling.

Implements [PhysicalObject](#).

Definition at line 31 of file CappedRectangle.h.

References [PhysicalObject::detectContacts\(\)](#).

4.8.3.8 **bool CappedRectangle::detectMouseContact (const [Vector2](#) & *rMouse*, [Vector2](#) & *p*, [PhysicalObject](#) *& *object*)** [virtual]

Detects whether a click of the mouse at *rMouse* has touched the object. In case of contact, returns true and sets *p* to the relative vector between the center of the object and the contact point; *p* is expressed in the object reference frame because it rotates with the object while the object is dragged.

Reimplemented from [PhysicalObject](#).

Definition at line 618 of file `CappedRectangle.cpp`.

References `cosAlpha`, `Vector2::getModule()`, `l`, `real`, `Vector2::rotate()`, `Vector2::setXY()`, `sinAlpha`, `Vector2::x`, and `Vector2::y`.

4.8.3.9 **void CappedRectangle::draw ([GUI](#) * *gui*)** [virtual]

Draws the object on the user interface.

Implements [GraphicalObject](#).

Definition at line 683 of file `CappedRectangle.cpp`.

References `GUI::drawCappedRectangle()`, `l`, `R`, `GUI::setBrushColor()`, `GUI::setPenColor()`, `Vector2::x`, and `Vector2::y`.

4.8.3.10 **void CappedRectangle::drawSensors ([GUI](#) * *gui*)**

Draws the object's tactile sensors.

Definition at line 693 of file `CappedRectangle.cpp`.

References `GUI::drawLine()`, `l`, `M_PI`, `nSensorsLateral`, `R`, `real`, `Vector2::x`, and `Vector2::y`.

4.8.3.11 **int CappedRectangle::getISensorLateral ([real](#) *side*, [real](#) *e*)** [private]

Returns the index of the lateral sensor corresponding to a contact given by *e* and the side of the capsule (a positive value corresponds to the top side of the capsule, a negative value to the bottom side). The projection of the contact point on the capsule axis is at a distance *e* from the capsule center (*e* can be positive or negative).

Definition at line 72 of file `CappedRectangle.cpp`.

References `l`, and `nSensorsLateral`.

4.8.3.12 **int CappedRectangle::getISensorLateral ([Vector2](#) & *p*, [real](#) *e*)** [private]

Returns the index of the lateral sensor corresponding to a contact given by *e* and the vector *p* relative to the center of the capsule. The projection of the contact point on the capsule axis is at a distance *e* from the capsule center (*e* can be positive or negative).

Definition at line 67 of file `CappedRectangle.cpp`.

References `getLateralSide()`, and `real`.

Referenced by `detectContacts()`, and `setISensorLateral()`.

4.8.3.13 `int CappedRectangle::getISensorSemicircle (Vector2 & p, int sign) [private]`

Gets the index of a tactile sensor on a semicircle, given the relative vector p from the end of the capsule's axis to the contact point, and the sign (+ or -) that determines one of the 2 ends of the capsule's axis.

Definition at line 38 of file CappedRectangle.cpp.

References `M_PI`, `nSensorsLateral`, `nSensorsSemicircle`, `real`, `Vector2::x`, and `Vector2::y`.

Referenced by `detectContacts()`.

4.8.3.14 `real CappedRectangle::getLateralSide (Vector2 & p) [inline, private]`

Returns the side on which exists a contact given by the vector p relative to the center of the capsule. A positive returned value corresponds to the top side of the capsule, a negative value to the bottom side.

Definition at line 109 of file CappedRectangle.h.

References `real`, `sinAlpha`, `Vector2::x`, and `Vector2::y`.

Referenced by `getISensorLateral()`, and `setISensorLateral()`.

4.8.3.15 `void CappedRectangle::setISensorLateral (int & i1, int & i2, Vector2 & p, real e1, real e2) [private]`

Gets the index of a tactile sensor on a lateral of the capsule, given the side of the capsule determined by the normal component (relative to the capsule's axis) of the vector p , and the linear coordinates of the ends of the lateral contact on the capsule's axis. `sinAlpha` and `cosAlpha` are passed as they are already computed.

Definition at line 61 of file CappedRectangle.cpp.

References `getISensorLateral()`, `getLateralSide()`, and `real`.

Referenced by `detectContacts()`.

4.8.3.16 `void CappedRectangle::setSensor (ContactInfo * contact) [virtual]`

Sets the activation of a tactile sensor, corresponding to the given contact. To be overridden.

Reimplemented from [PhysicalObject](#).

Definition at line 647 of file CappedRectangle.cpp.

References `ContactInfo::force`, `ContactInfo::iSensor`, `ContactInfo::iSensor2`, `real`, and `ContactInfo::type`.

4.8.4 Member Data Documentation**4.8.4.1** `real CappedRectangle::cosAlpha`

Cosinus of alpha. Always updated.

Definition at line 74 of file CappedRectangle.h.

Referenced by `computeBox()`, `VisualSensor::detectContacts()`, `detectContacts()`, and `detectMouseContact()`.

4.8.4.2 **real CappedRectangle::l**

Dimensions; l is the half length of the rectangle, R its half width. The end half circles will have radius R.

Definition at line 70 of file CappedRectangle.h.

Referenced by computeInertia(), computeMass(), VisualSensor::detectContacts(), detectContacts(), detectMouseContact(), draw(), drawSensors(), and getISensorLateral().

4.8.4.3 **unsigned CappedRectangle::nSensorsLateral**

The number of tactile sensors per lateral of the rectangle.

Definition at line 79 of file CappedRectangle.h.

Referenced by drawSensors(), getISensorLateral(), and getISensorSemicircle().

4.8.4.4 **unsigned CappedRectangle::nSensorsSemicircle**

The number of tactile sensors per end semicircle.

Definition at line 82 of file CappedRectangle.h.

Referenced by getISensorSemicircle().

4.8.4.5 **real CappedRectangle::R**

Definition at line 71 of file CappedRectangle.h.

Referenced by computeInertia(), computeMass(), VisualSensor::detectContacts(), detectContacts(), draw(), and drawSensors().

4.8.4.6 **real CappedRectangle::sinAlpha**

Sinus of alpha. Always updated.

Definition at line 76 of file CappedRectangle.h.

Referenced by computeBox(), VisualSensor::detectContacts(), detectContacts(), detectMouseContact(), and getLateralSide().

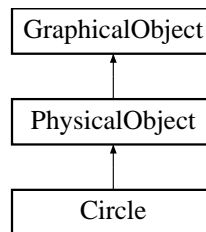
The documentation for this class was generated from the following files:

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

4.9 Circle Class Reference

```
#include <Circle.h>
```

Inheritance diagram for Circle::



Public Member Functions

- [Circle](#) ([real R](#), [real x=0](#), [real y=0](#), [std::string label=""](#), [int nSensors=0](#), [real sensorsStartAngle=0](#), [real saturationForce=0.5](#), [Color outlineColor=GUI::colorBlack](#), [Color fillColor=GUI::colorTransparent](#))
- virtual [~Circle](#) ()
- void [setRadius](#) ([real R](#))
- virtual void [computeMass](#) ([real density=ThyrixParameters::defaultDensity](#))
- virtual void [computeInertia](#) ()
- virtual int [isCircle](#) ()
- virtual bool [detectContacts](#) ([PhysicalObject *object](#), [GlobalContactInfoVector *contacts](#))
- virtual bool [detectContacts](#) ([Border *border](#), [GlobalContactInfoVector *contacts](#))
- virtual bool [detectContacts](#) ([Circle *circle](#), [GlobalContactInfoVector *contacts](#))
- virtual bool [detectMouseContact](#) ([const Vector2 &rMouse](#), [Vector2 &p](#), [PhysicalObject *&object](#))
- virtual void [computeBox](#) ()
- int [getISensor](#) ([Vector2 &p](#))
- virtual void [setSensor](#) ([ContactInfo *contact](#))
- virtual void [draw](#) ([GUI *gui](#))
- void [drawSensors](#) ([GUI *gui](#))

Public Attributes

- [real R](#)
- [real sensorsStartAngle](#)

4.9.1 Detailed Description

A circle. It has radius R.

Definition at line 15 of file Circle.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 `Circle::Circle (real R, real x = 0, real y = 0, std::string label = "", int nSensors = 0, real sensorsStartAngle = 0, real saturationForce = 0.5, Color outlineColor = GUI::colorBlack, Color fillColor = GUI::colorTransparent)`

Constructor, sets the position and radius, the tactile sensors (if any), computes the mass and inertia based on default density.

Definition at line 7 of file Circle.cpp.

References `computeBox()`, `computeInertia()`, `computeMass()`, `Vector2::x`, and `Vector2::y`.

4.9.2.2 `Circle::~~Circle ()` [virtual]

Destructor, does nothing.

Definition at line 22 of file Circle.cpp.

4.9.3 Member Function Documentation

4.9.3.1 `virtual void Circle::computeBox ()` [inline, virtual]

Computes the bounding box.

Reimplemented from [PhysicalObject](#).

Definition at line 62 of file Circle.h.

References `real`.

Referenced by `Circle()`, and `setRadius()`.

4.9.3.2 `void Circle::computeInertia ()` [virtual]

Computes the moment of inertia around the z axis that passes through the center, based on the mass and the radius.

Reimplemented from [PhysicalObject](#).

Definition at line 38 of file Circle.cpp.

References `R`.

Referenced by `Circle()`, and `setRadius()`.

4.9.3.3 `void Circle::computeMass (real density = ThyrixParameters::defaultDensity)`
[virtual]

Sets the mass based on the radius and the density.

Reimplemented from [PhysicalObject](#).

Definition at line 34 of file Circle.cpp.

References `R`.

Referenced by `Circle()`, and `setRadius()`.

4.9.3.4 `bool Circle::detectContacts (Circle * circle, GlobalContactInfoVector * contacts)` [virtual]

Tests for contacts between this object and a circle.

Reimplemented from [PhysicalObject](#).

Definition at line 81 of file Circle.cpp.

References [getISensor\(\)](#), [M_PI](#), [Vector2::normalize\(\)](#), [R](#), [PhysicalObject::r](#), [real](#), [sqr\(\)](#), [Vector2::x](#), and [Vector2::y](#).

4.9.3.5 `bool Circle::detectContacts (Border * border, GlobalContactInfoVector * contacts)` [virtual]

Tests for contacts between this object and a border.

Reimplemented from [PhysicalObject](#).

Definition at line 43 of file Circle.cpp.

References [getISensor\(\)](#), [Border::normal](#), [PhysicalObject::r](#), [real](#), [Vector2::setToZero\(\)](#), [Vector2::x](#), and [Vector2::y](#).

4.9.3.6 `virtual bool Circle::detectContacts (PhysicalObject * object, GlobalContactInfoVector * contacts)` [inline, virtual]

Redirects contact detection.

Implements [PhysicalObject](#).

Definition at line 45 of file Circle.h.

References [PhysicalObject::detectContacts\(\)](#).

Referenced by [Spherus::detectContacts\(\)](#), and [Spherus::detectInternalContacts\(\)](#).

4.9.3.7 `bool Circle::detectMouseContact (const Vector2 & rMouse, Vector2 & p, PhysicalObject *& object)` [virtual]

Detects whether a click of the mouse at [rMouse](#) has touched the object. In case of contact, returns true and sets [p](#) to the relative vector between the center of the object and the contact point; [p](#) is expressed in the object reference frame because it rotates with the object while the object is dragged.

Reimplemented from [PhysicalObject](#).

Definition at line 121 of file Circle.cpp.

References [Vector2::getSquaredModule\(\)](#), [R](#), and [Vector2::rotate\(\)](#).

Referenced by [Spherus::detectMouseContact\(\)](#).

4.9.3.8 `void Circle::draw (GUI * gui)` [virtual]

Draws the circle on the user interface.

Implements [GraphicalObject](#).

Definition at line 160 of file Circle.cpp.

References `GUI::drawCircle()`, `R`, `GUI::setBrushColor()`, `GUI::setPenColor()`, `Vector2::x`, and `Vector2::y`.
Referenced by `Spherus::draw()`.

4.9.3.9 void Circle::drawSensors (GUI * gui)

Draws the activation of the circle's contact sensors.

Definition at line 170 of file `Circle.cpp`.

References `GUI::drawLine()`, `M_PI`, `R`, `real`, `sensorsStartAngle`, `Vector2::x`, and `Vector2::y`.

4.9.3.10 int Circle::getISensor (Vector2 & p)

Returns the index of the tactile sensor activated by a contact at relative position `p`.

Definition at line 137 of file `Circle.cpp`.

References `real`, `Vector2::x`, and `Vector2::y`.

Referenced by `detectContacts()`, and `CappedRectangle::detectContacts()`.

4.9.3.11 virtual int Circle::isCircle () [inline, virtual]

Tells that this object has circular symmetry.

Reimplemented from [PhysicalObject](#).

Definition at line 42 of file `Circle.h`.

4.9.3.12 void Circle::setRadius (real R)

Sets the radius of the circle, computes the mass and inertia based on default density.

Definition at line 27 of file `Circle.cpp`.

References `computeBox()`, `computeInertia()`, and `computeMass()`.

4.9.3.13 void Circle::setSensor (ContactInfo * contact) [virtual]

Sets the activation of a tactile sensor, corresponding to the given contact. To be overridden.

Reimplemented from [PhysicalObject](#).

Definition at line 151 of file `Circle.cpp`.

References `ContactInfo::force`, `ContactInfo::iSensor`, and `real`.

4.9.4 Member Data Documentation

4.9.4.1 real Circle::R

Radius of the circle

Definition at line 29 of file `Circle.h`.

Referenced by `computeInertia()`, `computeMass()`, `Iunctus::controll()`, `VisualSensor::detectContacts()`, `detectContacts()`, `CappedRectangle::detectContacts()`, `detectMouseContact()`, `Iunctus::draw()`, `draw()`, and `drawSensors()`.

4.9.4.2 `real Circle::sensorsStartAngle`

The start angle for the contact sensors; $0..2\pi$

Definition at line 69 of file `Circle.h`.

Referenced by `drawSensors()`.

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

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

4.10 Color Class Reference

```
#include <Color.h>
```

Public Member Functions

- [Color](#) (unsigned char *r*=0, unsigned char *g*=0, unsigned char *b*=0, bool *transparent*=false)
- [Color](#) (const char *name)
- virtual [~Color](#) ()
- void [setTransparent](#) ()

Public Attributes

- unsigned char *r*
- unsigned char *g*
- unsigned char *b*
- bool *transparent*

Static Private Attributes

- [ColorDefinitions](#) *definitions*

4.10.1 Constructor & Destructor Documentation

4.10.1.1 [Color::Color](#) (unsigned char *r* = 0, unsigned char *g* = 0, unsigned char *b* = 0, bool *transparent* = false)

Definition at line 14 of file [Color.cpp](#).

4.10.1.2 [Color::Color](#) (const char * *name*)

Definition at line 18 of file [Color.cpp](#).

References [b](#), [definitions](#), [g](#), [r](#), [ColorDefinitions::setColor\(\)](#), and [transparent](#).

4.10.1.3 [Color::~Color](#) () [virtual]

Definition at line 25 of file [Color.cpp](#).

4.10.2 Member Function Documentation

4.10.2.1 void [Color::setTransparent](#) () [inline]

Definition at line 18 of file [Color.h](#).

4.10.3 Member Data Documentation

4.10.3.1 unsigned char [Color::b](#)

Definition at line 15 of file Color.h.

Referenced by Color(), GUIWx::setBrushColor(), and GUIWx::setPenColor().

4.10.3.2 [ColorDefinitions Color::definitions](#) [static, private]

Definition at line 12 of file Color.cpp.

Referenced by Color().

4.10.3.3 unsigned char [Color::g](#)

Definition at line 15 of file Color.h.

Referenced by Color(), GUIWx::setBrushColor(), and GUIWx::setPenColor().

4.10.3.4 unsigned char [Color::r](#)

Definition at line 15 of file Color.h.

Referenced by Color(), GUIWx::setBrushColor(), and GUIWx::setPenColor().

4.10.3.5 bool [Color::transparent](#)

Definition at line 16 of file Color.h.

Referenced by Color(), GUIWx::setBrushColor(), and GUIWx::setPenColor().

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

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

4.11 ColorDefinitions Class Reference

```
#include <ColorDefinitions.h>
```

Public Member Functions

- [ColorDefinitions](#) ()
- virtual [~ColorDefinitions](#) ()
- void [setColor](#) (unsigned char &r, unsigned char &g, unsigned char &b, const char *name)

Private Attributes

- std::map< const char *, [SimpleColor](#), [ltstr](#) > [names](#)

4.11.1 Constructor & Destructor Documentation

4.11.1.1 ColorDefinitions::ColorDefinitions ()

Definition at line 11 of file ColorDefinitions.cpp.

References [names](#).

4.11.1.2 ColorDefinitions::~~ColorDefinitions () [virtual]

Definition at line 90 of file ColorDefinitions.cpp.

4.11.2 Member Function Documentation

4.11.2.1 void ColorDefinitions::setColor (unsigned char & *r*, unsigned char & *g*, unsigned char & *b*, const char * *name*)

Definition at line 94 of file ColorDefinitions.cpp.

References [names](#).

Referenced by [Color::Color\(\)](#).

4.11.3 Member Data Documentation

4.11.3.1 std::map<const char*, [SimpleColor](#), [ltstr](#)> [ColorDefinitions::names](#) [private]

Definition at line 31 of file ColorDefinitions.h.

Referenced by [ColorDefinitions\(\)](#), and [setColor\(\)](#).

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

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

4.12 ColorDefinitions::ltstr Struct Reference

Public Member Functions

- bool [operator\(\)](#) (const char *s1, const char *s2) const

4.12.1 Member Function Documentation

4.12.1.1 bool ColorDefinitions::ltstr::operator() (const char * s1, const char * s2) const [inline]

Definition at line 26 of file ColorDefinitions.h.

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

- [ColorDefinitions.h](#)

4.13 ColorDefinitions::SimpleColor Struct Reference

Public Member Functions

- [SimpleColor](#) ()
- [SimpleColor](#) (unsigned char x, unsigned char y, unsigned char z)

Public Attributes

- unsigned char [r](#)
- unsigned char [g](#)
- unsigned char [b](#)

4.13.1 Constructor & Destructor Documentation

4.13.1.1 ColorDefinitions::SimpleColor::SimpleColor () [inline]

Definition at line 16 of file ColorDefinitions.h.

4.13.1.2 ColorDefinitions::SimpleColor::SimpleColor (unsigned char x, unsigned char y, unsigned char z) [inline]

Definition at line 19 of file ColorDefinitions.h.

4.13.2 Member Data Documentation

4.13.2.1 unsigned char [ColorDefinitions::SimpleColor::b](#)

Definition at line 22 of file ColorDefinitions.h.

4.13.2.2 unsigned char [ColorDefinitions::SimpleColor::g](#)

Definition at line 22 of file ColorDefinitions.h.

4.13.2.3 unsigned char [ColorDefinitions::SimpleColor::r](#)

Definition at line 22 of file ColorDefinitions.h.

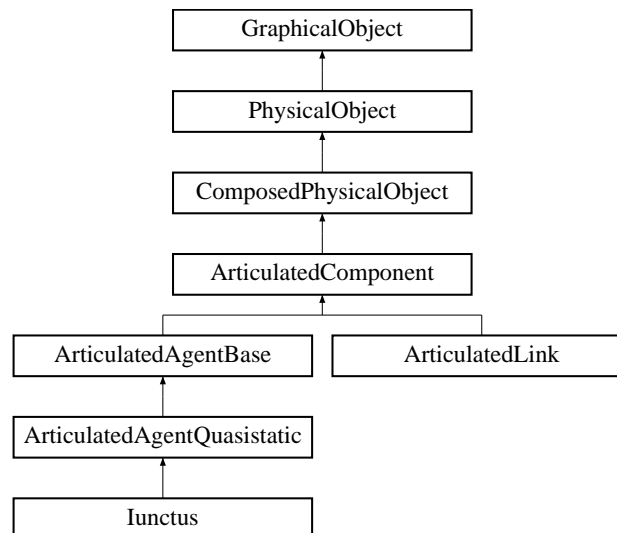
The documentation for this struct was generated from the following file:

- [ColorDefinitions.h](#)

4.14 ComposedPhysicalObject Class Reference

```
#include <ComposedPhysicalObject.h>
```

Inheritance diagram for ComposedPhysicalObject::



Public Member Functions

- [ComposedPhysicalObject](#) (std::string label="")
- [ComposedPhysicalObject](#) (real x, real y, real alpha=0, std::string label="")
- virtual [~ComposedPhysicalObject](#) ()
- void [addObject](#) ([PhysicalObject](#) *object, real x, real y, real alpha)
- void [deleteObjects](#) ()
- void [computeMassProperties](#) ()
- void [computeMemberPositions](#) ()
- virtual void [integrate](#) (const [Integrator](#) &integrator)
- virtual void [rollback](#) ()
- virtual void [registerPrimitives](#) ([Simulator](#) *simulator)
- virtual bool [detectContacts](#) ([PhysicalObject](#) *object, [GlobalContactInfoVector](#) *contacts)
- virtual bool [detectMouseContact](#) (const [Vector2](#) &rMouse, [Vector2](#) &p, [PhysicalObject](#) *&object)
- virtual void [setSensors](#) ()
- virtual void [resetSensors](#) ()
- void [draw](#) ([GUI](#) *gui)
- void [setOutlineColor](#) ([Color](#) color)
- void [setFillColor](#) ([Color](#) color)

Public Attributes

- [Vector2](#) s
- [PhysicalObjectPVector](#) objects

4.14.1 Detailed Description

A composed object can consist of several physical objects (e.g., several spheres along a line segment) that are rigidly bind together.

A composed object has a reference point that is not necessarily its center of mass. The *r*, *v* parameters refer to the position of the reference point relative to the laboratory reference system (LRS). *m* is the total mass of the composed object. *alpha*, *omega*, *I* are properties of the rotation around the reference point.

The composed object owns the member objects (will destroy them on destruction).

Definition at line 23 of file ComposedPhysicalObject.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 ComposedPhysicalObject::ComposedPhysicalObject (std::string *label* = "")

Definition at line 6 of file ComposedPhysicalObject.cpp.

4.14.2.2 ComposedPhysicalObject::ComposedPhysicalObject (real *x*, real *y*, real *alpha* = 0, std::string *label* = "")

Definition at line 10 of file ComposedPhysicalObject.cpp.

References Vector2::setXY().

4.14.2.3 ComposedPhysicalObject::~ComposedPhysicalObject () [virtual]

Deletes member objects.

Definition at line 16 of file ComposedPhysicalObject.cpp.

References deleteObjects().

4.14.3 Member Function Documentation

4.14.3.1 void ComposedPhysicalObject::addObject (PhysicalObject * *object*, real *x*, real *y*, real *alpha*)

Adds a previously created object to the objects composing the link / composed object, also computes (updates) the mass properties.

Parameters:

object is a previously created object

x and

y specifies the position of the center of mass of the added object relative to the composed object reference system (ORS)

alpha specifies the (fixed) rotation angle of the added object relative to the ORS

Definition at line 22 of file ComposedPhysicalObject.cpp.

References computeMassProperties(), PhysicalObject::label, objects, PhysicalObject::parent, PhysicalObject::relativeAlpha, PhysicalObject::relativeR, and Vector2::setXY().

Referenced by `ArticulatedLimb::addOneObjectLink()`, `Iunctus::build()`, and `IunctusSimulator::IunctusSimulator()`.

4.14.3.2 void ComposedPhysicalObject::computeMassProperties ()

Computes s , the position of the center of mass of the composed object; the total mass of the composed object; and the moment of inertia around its reference point.

Definition at line 36 of file `ComposedPhysicalObject.cpp`.

References `Vector2::getSquaredModule()`, `PhysicalObject::I`, `PhysicalObject::m`, `objects`, `PhysicalObject::relativeR`, s , and `Vector2::setToZero()`.

Referenced by `addObject()`.

4.14.3.3 void ComposedPhysicalObject::computeMemberPositions ()

Computes the absolute positions and angles of the member objects (positions relative to the LRS) based on the position of the ORS relative to LRS and the positions of the member objects relative to the ORS. Also computes the bounding box.

Definition at line 55 of file `ComposedPhysicalObject.cpp`.

References `PhysicalObject::alpha`, `PhysicalObject::alphaOld`, `PhysicalObject::boxMax`, `PhysicalObject::boxMin`, `PhysicalObject::computeBox()`, `objects`, `PhysicalObject::r`, `PhysicalObject::relativeAlpha`, `PhysicalObject::relativeR`, `PhysicalObject::rOld`, `Vector2::rotate()`, `Vector2::setXY()`, `Vector2::updateMax()`, and `Vector2::updateMin()`.

Referenced by `Iunctus::build()`, `ArticulatedAgentQuasistatic::forwardKinematics()`, `integrate()`, and `rollback()`.

4.14.3.4 void ComposedPhysicalObject::deleteObjects ()

Deletes member objects and empties the corresponding list.

Definition at line 31 of file `ComposedPhysicalObject.cpp`.

References `objects`, and `purgeContainer()`.

Referenced by `~ComposedPhysicalObject()`.

4.14.3.5 bool ComposedPhysicalObject::detectContacts ([PhysicalObject](#) * *object*, [GlobalContactInfoVector](#) * *contacts*) [virtual]

Contact handling.

Implements [PhysicalObject](#).

Reimplemented in [ArticulatedAgentBase](#).

Definition at line 83 of file `ComposedPhysicalObject.cpp`.

References `Simulator::detectContacts()`, and `objects`.

Referenced by `ArticulatedAgentBase::detectContacts()`, and `ArticulatedAgentBase::detectInternalContacts()`.

4.14.3.6 **bool ComposedPhysicalObject::detectMouseContact** (const **Vector2** & *rMouse*, **Vector2** & *p*, **PhysicalObject** *& *object*) [virtual]

Detects whether a click of the mouse at *rMouse* has touched the object. In case of contact, returns true and sets *p* to the relative vector between the center of the composed object and the contact point; *p* is expressed in the composed object reference frame because it rotates with the object while the object is dragged.

Reimplemented from [PhysicalObject](#).

Reimplemented in [ArticulatedAgentBase](#).

Definition at line 91 of file `ComposedPhysicalObject.cpp`.

References `objects`, and `Vector2::rotate()`.

Referenced by `ArticulatedAgentBase::detectMouseContact()`.

4.14.3.7 **void ComposedPhysicalObject::draw** (**GUI** * *gui*) [virtual]

Calls the draw method for the member objects.

Implements [GraphicalObject](#).

Reimplemented in [Iunctus](#), [ArticulatedAgentBase](#), and [ArticulatedLink](#).

Definition at line 123 of file `ComposedPhysicalObject.cpp`.

References `GraphicalObject::draw()`, and `objects`.

Referenced by `ArticulatedLink::draw()`, and `ArticulatedAgentBase::draw()`.

4.14.3.8 **virtual void ComposedPhysicalObject::integrate** (const **Integrator** & *integrator*) [inline, virtual]

Advances the time to the next timestep.

Reimplemented from [PhysicalObject](#).

Reimplemented in [ArticulatedAgentQuasistatic](#), and [ArticulatedLink](#).

Definition at line 63 of file `ComposedPhysicalObject.h`.

References `computeMemberPositions()`, and `PhysicalObject::integrate()`.

Referenced by `ArticulatedAgentQuasistatic::integrate()`.

4.14.3.9 **void ComposedPhysicalObject::registerPrimitives** (**Simulator** * *simulator*) [virtual]

Registers all composing primitives of the object with the simulator.

Reimplemented from [PhysicalObject](#).

Reimplemented in [ArticulatedAgentBase](#).

Definition at line 76 of file `ComposedPhysicalObject.cpp`.

References `objects`, and `Simulator::registerPrimitive()`.

Referenced by `ArticulatedAgentBase::registerPrimitives()`.

4.14.3.10 void ComposedPhysicalObject::resetSensors () [virtual]

Resets tactile sensors to zero.

Reimplemented from [PhysicalObject](#).

Definition at line 116 of file ComposedPhysicalObject.cpp.

References [objects](#), and [PhysicalObject::resetSensors\(\)](#).

4.14.3.11 virtual void ComposedPhysicalObject::rollback () [inline, virtual]

Rolls back the time to the previous timestep.

Reimplemented from [PhysicalObject](#).

Reimplemented in [ArticulatedAgentQuasistatic](#), and [ArticulatedLink](#).

Definition at line 69 of file ComposedPhysicalObject.h.

References [computeMemberPositions\(\)](#), and [PhysicalObject::rollback\(\)](#).

Referenced by [ArticulatedLink::rollback\(\)](#), and [ArticulatedAgentQuasistatic::rollback\(\)](#).

4.14.3.12 void ComposedPhysicalObject::setFillColor ([Color](#) color) [virtual]

Sets fill color.

Reimplemented from [GraphicalObject](#).

Reimplemented in [ArticulatedAgentBase](#).

Definition at line 139 of file ComposedPhysicalObject.cpp.

References [objects](#).

Referenced by [ArticulatedLimb::setFillColor\(\)](#), and [ArticulatedAgentBase::setFillColor\(\)](#).

4.14.3.13 void ComposedPhysicalObject::setOutlineColor ([Color](#) color) [virtual]

Sets outline color.

Reimplemented from [GraphicalObject](#).

Reimplemented in [ArticulatedAgentBase](#).

Definition at line 131 of file ComposedPhysicalObject.cpp.

References [objects](#).

Referenced by [ArticulatedLimb::setOutlineColor\(\)](#), and [ArticulatedAgentBase::setOutlineColor\(\)](#).

4.14.3.14 void ComposedPhysicalObject::setSensors () [virtual]

Sets the activations of the tactile sensors, on the basis of the contact forces related to the contacts of the object with other objects.

Reimplemented from [PhysicalObject](#).

Definition at line 105 of file ComposedPhysicalObject.cpp.

References [ContactInfo::iSensor](#), [ContactInfo::object](#), and [PhysicalObject::setSensor\(\)](#).

Referenced by `ArticulatedAgentQuasistatic::computeDerivatives()`.

4.14.4 Member Data Documentation

4.14.4.1 [PhysicalObjectPVector](#) [ComposedPhysicalObject::objects](#)

The list of member objects. The list owns the member objects.

Definition at line 96 of file `ComposedPhysicalObject.h`.

Referenced by `addObject()`, `computeMassProperties()`, `computeMemberPositions()`, `deleteObjects()`, `detectContacts()`, `detectMouseContact()`, `draw()`, `registerPrimitives()`, `resetSensors()`, `setFillColor()`, and `setOutlineColor()`.

4.14.4.2 [Vector2](#) [ComposedPhysicalObject::s](#)

Position of the center of mass relative to the reference point (in the object reference system, ORS).

Definition at line 34 of file `ComposedPhysicalObject.h`.

Referenced by `computeMassProperties()`.

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

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

4.15 ContactInfo Class Reference

```
#include <ContactInfo.h>
```

Public Member Functions

- [ContactInfo](#) ([PhysicalObject](#) *object, int sigma)
- [ContactInfo](#) ([PhysicalObject](#) *object, int sigma, const [Vector2](#) &p, const [Vector2](#) &n, int iSensor=-1)
- [ContactInfo](#) ([PhysicalObject](#) *object, int sigma, const [Vector2](#) &p, const [Vector2](#) &n, int iSensor, int iSensor2)
- virtual [~ContactInfo](#) ()

Public Attributes

- int sigma
- [ContactType](#) type
- [Vector2](#) p
- [Vector2](#) n
- real pxn
- int iSensor
- int iSensor2
- [PhysicalObject](#) * parentObject
- [PhysicalObject](#) * object
- real force
- int alpha

Private Member Functions

- void [setupObject](#) ()

4.15.1 Constructor & Destructor Documentation

4.15.1.1 [ContactInfo::ContactInfo](#) ([PhysicalObject](#) * object, int sigma)

Constructs a torque contact.

Definition at line 5 of file [ContactInfo.cpp](#).

References [contactTypeTorque](#), and [parentObject](#).

4.15.1.2 [ContactInfo::ContactInfo](#) ([PhysicalObject](#) * object, int sigma, const [Vector2](#) & p, const [Vector2](#) & n, int iSensor = -1)

Constructs a force contact. Initializes information, and also adds the contact to the object (or its parent, if exists - for composed objects)

Definition at line 13 of file [ContactInfo.cpp](#).

References [contactTypeForce](#), and [setupObject\(\)](#).

4.15.1.3 **ContactInfo::ContactInfo** (**PhysicalObject** * *object*, int *sigma*, const **Vector2** & *p*, const **Vector2** & *n*, int *iSensor*, int *iSensor2*)

Constructs a force contact for parallel contact of capsules.

Definition at line 26 of file ContactInfo.cpp.

References contactTypeForceParallel, and setupObject().

4.15.1.4 **ContactInfo::~ContactInfo** () [virtual]

Definition at line 59 of file ContactInfo.cpp.

4.15.2 Member Function Documentation

4.15.2.1 **void ContactInfo::setupObject** () [private]

Definition at line 40 of file ContactInfo.cpp.

References **PhysicalObject::addContact**(), *object*, *p*, **PhysicalObject::parent**, **parentObject**, **pxn**, and **PhysicalObject::r**.

Referenced by **ContactInfo**().

4.15.3 Member Data Documentation

4.15.3.1 **int ContactInfo::alpha**

The index of this contact in the global contact vector.

Definition at line 68 of file ContactInfo.h.

Referenced by **ArticulatedAgentQuasistatic::backwardDynamics**(), **PhysicalObject::fillContactMatrix**(), **ArticulatedAgentBase::fillContactMatrix**(), and **ArticulatedAgentQuasistatic::forwardAccelerations**().

4.15.3.2 **real ContactInfo::force**

The modulus of the contact force.

Definition at line 65 of file ContactInfo.h.

Referenced by **PhysicalObject::computeDerivatives**(), **PhysicalObject::drawContactForces**(), **Circle::setSensor**(), and **CappedRectangle::setSensor**().

4.15.3.3 **int ContactInfo::iSensor**

The index of the tactile sensor, or of the ends of a range of tactile sensors (for parallel contact of capsules) affected by the contact, if any.

Definition at line 55 of file ContactInfo.h.

Referenced by **Circle::setSensor**(), **CappedRectangle::setSensor**(), and **ComposedPhysicalObject::setSensors**().

4.15.3.4 int ContactInfo::iSensor2

The index of the tactile sensor, or of the ends of a range of tactile sensors (for parallel contact of capsules) affected by the contact, if any.

Definition at line 55 of file ContactInfo.h.

Referenced by CappedRectangle::setSensor().

4.15.3.5 Vector2 ContactInfo::n

The direction of the contact force (the normal to the contact plane), in the laboratory reference system.

Definition at line 48 of file ContactInfo.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), PhysicalObject::computeDerivatives(), PhysicalObject::drawContactForces(), PhysicalObject::fillContactMatrix(), ArticulatedAgentBase::fillContactMatrix(), and ArticulatedAgentQuasistatic::forwardAccelerations().

4.15.3.6 PhysicalObject* ContactInfo::object

The object to which the contact belongs.

Definition at line 62 of file ContactInfo.h.

Referenced by ComposedPhysicalObject::setSensors(), and setupObject().

4.15.3.7 Vector2 ContactInfo::p

The position of the contact point relative to the object, in the laboratory reference system.

Definition at line 45 of file ContactInfo.h.

Referenced by PhysicalObject::drawContactForces(), PhysicalObject::fillContactMatrix(), ArticulatedAgentBase::fillContactMatrix(), ArticulatedAgentQuasistatic::forwardAccelerations(), and setupObject().

4.15.3.8 PhysicalObject* ContactInfo::parentObject

The object that processes the contact (for objects that are part of a composed object, the composed object processes the contact).

Definition at line 59 of file ContactInfo.h.

Referenced by ContactInfo(), and setupObject().

4.15.3.9 real ContactInfo::pxn

The cross product $\mathbf{p} \times \mathbf{n}$ (having only a component on the z axis).

Definition at line 50 of file ContactInfo.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), PhysicalObject::computeDerivatives(), PhysicalObject::fillContactMatrix(), and setupObject().

4.15.3.10 int [ContactInfo::sigma](#)

The sign of the participation of this object to the contact

Definition at line 37 of file [ContactInfo.h](#).

Referenced by [ArticulatedAgentQuasistatic::backwardDynamics\(\)](#), [PhysicalObject::computeDerivatives\(\)](#), [PhysicalObject::drawContactForces\(\)](#), [PhysicalObject::fillContactMatrix\(\)](#), [ArticulatedAgentBase::fillContactMatrix\(\)](#), and [ArticulatedAgentQuasistatic::forwardAccelerations\(\)](#).

4.15.3.11 [ContactType](#) [ContactInfo::type](#)

The type of the contact: force or torque.

Definition at line 40 of file [ContactInfo.h](#).

Referenced by [CappedRectangle::setSensor\(\)](#).

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

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

4.16 ContactSolver Class Reference

```
#include <ContactSolver.h>
```

Public Member Functions

- [ContactSolver](#) ([GlobalContactInfoVector](#) *contacts)
- virtual [~ContactSolver](#) ()
- void [init](#) ()
- void [computeContacts](#) ()
- void [computeContactsPreda](#) ()
- void [uploadForces](#) ()

Public Attributes

- [real](#) ** [contactMatrix](#)
- [real](#) * [contactVector](#)
- [real](#) * [tempVector1](#)
- [real](#) * [tempVector2](#)
- [real](#) ** [tempMatrix](#)

Private Member Functions

- [real](#) [maxStep](#) (int d, int *sIndex)
- bool [driveToZero](#) (int d)
- void [initForcesVelocities](#) ()

Private Attributes

- [GlobalContactInfoVector](#) * [contacts](#)
- int [nContacts](#)
- int [nContactsMax](#)
- [real](#) ** [contactMatrixTemp](#)
- [real](#) * [contactForces](#)
- [real](#) * [contactVectorTemp](#)
- [real](#) * [contactDForces](#)
- [real](#) * [contactVelocities](#)
- [ContactSet](#) * [contactSets](#)
- int * [contactIndexC](#)
- int * [contactIndexTemp](#)
- int * [zeros](#)

4.16.1 Constructor & Destructor Documentation

4.16.1.1 `ContactSolver::ContactSolver (GlobalContactInfoVector * contacts)`

Definition at line 4 of file `ContactSolver.cpp`.

References `contactDForces`, `contactForces`, `contactIndexC`, `contactIndexTemp`, `contactMatrix`, `contactMatrixTemp`, `contacts`, `contactSets`, `contactVector`, `contactVectorTemp`, `contactVelocities`, `nContacts`, `nContactsMax`, `tempMatrix`, `tempVector1`, `tempVector2`, and `zeros`.

4.16.1.2 `ContactSolver::~ContactSolver ()` [virtual]

Definition at line 26 of file `ContactSolver.cpp`.

References `contactMatrix`, `contactMatrixTemp`, and `tempMatrix`.

4.16.2 Member Function Documentation

4.16.2.1 `void ContactSolver::computeContacts ()`

Computes the contact forces, using the Baraff algorithm.

Definition at line 273 of file `ContactSolver.cpp`.

References `computeContactsPreda()`, `driveToZero()`, `initForcesVelocities()`, and `nContacts`.

Referenced by `Simulator::advanceTime()`, and `Simulator::computeContacts()`.

4.16.2.2 `void ContactSolver::computeContactsPreda ()`

Computes the contact forces with a Gauss-Seidel like method (re)discovered by Preda.

Definition at line 308 of file `ContactSolver.cpp`.

References `contactForces`, `contactMatrix`, `contactVector`, `initForcesVelocities()`, and `real`.

Referenced by `computeContacts()`.

4.16.2.3 `bool ContactSolver::driveToZero (int d)` [private]

Definition at line 151 of file `ContactSolver.cpp`.

References `contactDForces`, `contactForces`, `contactIndexC`, `contactMatrix`, `contactMatrixTemp`, `contactSets`, `contactVectorTemp`, `contactVelocities`, `maxStep()`, `nContacts`, `real`, `SystemSolver::svDecompose()`, `SystemSolver::svSubstitute()`, `tempMatrix`, `tempVector1`, `tempVector2`, and `zeros`.

Referenced by `computeContacts()`.

4.16.2.4 `void ContactSolver::init ()`

Resize the data members to allow a greater number of contacts, if needed; set to zero `contactMatrix`, `contactVector`, `contactForces`.

Definition at line 50 of file `ContactSolver.cpp`.

References `contactDForces`, `contactForces`, `contactIndexC`, `contactIndexTemp`, `contactMatrix`, `contactMatrixTemp`, `contacts`, `ContactSet`, `contactSets`, `contactVector`, `contactVectorTemp`, `contactVelocities`, `nContacts`, `nContactsMax`, `real`, `tempMatrix`, `tempVector1`, `tempVector2`, and `zeros`.

Referenced by `Simulator::indexContacts()`.

4.16.2.5 `void ContactSolver::initForcesVelocities ()` [private]

Definition at line 262 of file `ContactSolver.cpp`.

References `contactForces`, `contactVector`, `contactVelocities`, and `zeros`.

Referenced by `computeContacts()`, and `computeContactsPreda()`.

4.16.2.6 `real ContactSolver::maxStep (int d, int * sIndex)` [private]

Definition at line 107 of file `ContactSolver.cpp`.

References `contactDForces`, `contactForces`, `contactSets`, `contactVectorTemp`, `contactVelocities`, and `real`.

Referenced by `driveToZero()`.

4.16.2.7 `void ContactSolver::uploadForces ()`

Uploads the computed forces into simulator contacts.

Definition at line 344 of file `ContactSolver.cpp`.

References `contactForces`, and `contacts`.

Referenced by `Simulator::advanceTime()`.

4.16.3 Member Data Documentation

4.16.3.1 `real* ContactSolver::contactDForces` [private]

The delta f vector of the Baraff algorithm

Definition at line 67 of file `ContactSolver.h`.

Referenced by `ContactSolver()`, `driveToZero()`, `init()`, and `maxStep()`.

4.16.3.2 `real* ContactSolver::contactForces` [private]

The f vector of the Baraff algorithm.

Definition at line 61 of file `ContactSolver.h`.

Referenced by `computeContactsPreda()`, `ContactSolver()`, `driveToZero()`, `init()`, `initForcesVelocities()`, `maxStep()`, and `uploadForces()`.

4.16.3.3 `int* ContactSolver::contactIndexC` [private]

A index vector used in the Baraff algorithm to point to C contacts.

Definition at line 76 of file `ContactSolver.h`.

Referenced by ContactSolver(), driveToZero(), and init().

4.16.3.4 **int*** **ContactSolver::contactIndexTemp** [private]

A temporary index vector used in the Baraff algorithm for system solving.

Definition at line 82 of file ContactSolver.h.

Referenced by ContactSolver(), and init().

4.16.3.5 **real**** **ContactSolver::contactMatrix**

The A matrix of the Baraff algorithm.

Definition at line 33 of file ContactSolver.h.

Referenced by computeContactsPreda(), ContactSolver(), driveToZero(), PhysicalObject::fillContactMatrix(), ArticulatedAgentBase::fillContactMatrix(), ArticulatedAgentQuasistatic::forwardAccelerations(), init(), and ~ContactSolver().

4.16.3.6 **real**** **ContactSolver::contactMatrixTemp** [private]

A temporary matrix used in the Baraff algorithm for holding the A_CC.

Definition at line 58 of file ContactSolver.h.

Referenced by ContactSolver(), driveToZero(), init(), and ~ContactSolver().

4.16.3.7 **GlobalContactInfoVector*** **ContactSolver::contacts** [private]

A pointer to the contacts list.

Definition at line 49 of file ContactSolver.h.

Referenced by ContactSolver(), init(), and uploadForces().

4.16.3.8 **ContactSet*** **ContactSolver::contactSets** [private]

A vector which holds the set to which a contact belongs (C or NC) in the Baraff algorithm

Definition at line 73 of file ContactSolver.h.

Referenced by ContactSolver(), driveToZero(), init(), and maxStep().

4.16.3.9 **real*** **ContactSolver::contactVector**

The b vector of the Baraff algorithm.

Definition at line 36 of file ContactSolver.h.

Referenced by computeContactsPreda(), ContactSolver(), PhysicalObject::fillContactMatrix(), ArticulatedAgentBase::fillContactMatrix(), ArticulatedAgentQuasistatic::forwardAccelerations(), init(), and initForcesVelocities().

4.16.3.10 `real* ContactSolver::contactVectorTemp` [private]

A temporary vector used in the Baraff algorithm.

Definition at line 64 of file ContactSolver.h.

Referenced by ContactSolver(), driveToZero(), init(), and maxStep().

4.16.3.11 `real* ContactSolver::contactVelocities` [private]

The a vector of the Baraff algorithm.

Definition at line 70 of file ContactSolver.h.

Referenced by ContactSolver(), driveToZero(), init(), initForcesVelocities(), and maxStep().

4.16.3.12 `int ContactSolver::nContacts` [private]

The current number of contacts.

Definition at line 52 of file ContactSolver.h.

Referenced by computeContacts(), ContactSolver(), driveToZero(), and init().

4.16.3.13 `int ContactSolver::nContactsMax` [private]

The maximum number of contacts that fit in the current contact data structures.

Definition at line 55 of file ContactSolver.h.

Referenced by ContactSolver(), and init().

4.16.3.14 `real** ContactSolver::tempMatrix`

Definition at line 41 of file ContactSolver.h.

Referenced by ContactSolver(), driveToZero(), init(), and ~ContactSolver().

4.16.3.15 `real* ContactSolver::tempVector1`

A temporary vector for system solving.

Definition at line 39 of file ContactSolver.h.

Referenced by ContactSolver(), driveToZero(), and init().

4.16.3.16 `real* ContactSolver::tempVector2`

Definition at line 40 of file ContactSolver.h.

Referenced by ContactSolver(), driveToZero(), and init().

4.16.3.17 `int* ContactSolver::zeros` [private]

Counts the number of s=0 cycles for each contact

Definition at line 85 of file ContactSolver.h.

Referenced by `ContactSolver()`, `driveToZero()`, `init()`, and `initForcesVelocities()`.

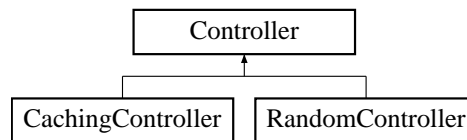
The documentation for this class was generated from the following files:

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

4.17 Controller Class Reference

```
#include <Controller.h>
```

Inheritance diagram for Controller::



Public Member Functions

- **Controller** (unsigned int [nInputs](#), unsigned int [nOutputs](#))
- virtual **~Controller** ()
- float **getOutput** (unsigned int i)
- void **setInput** (unsigned int i, float value)
- virtual void **advanceTime** ()

Public Attributes

- unsigned int [nInputs](#)
- unsigned int [nOutputs](#)

Protected Attributes

- float * [input](#)
- float * [output](#)

4.17.1 Constructor & Destructor Documentation

4.17.1.1 Controller::Controller (unsigned int *nInputs*, unsigned int *nOutputs*)

Definition at line 6 of file Controller.cpp.

References [input](#), and [output](#).

4.17.1.2 Controller::~~Controller () [virtual]

Definition at line 17 of file Controller.cpp.

4.17.2 Member Function Documentation

4.17.2.1 virtual void Controller::advanceTime () [inline, virtual]

Reimplemented in [RandomController](#).

Definition at line 21 of file Controller.h.

Referenced by `Spherus::controll()`.

4.17.2.2 `float Controller::getOutput (unsigned int i)` `[inline]`

Definition at line 11 of file `Controller.h`.

Referenced by `Spherus::controll()`, `Pac::controll()`, and `Iunctus::controll()`.

4.17.2.3 `void Controller::setInput (unsigned int i, float value)` `[inline]`

Definition at line 16 of file `Controller.h`.

Referenced by `Spherus::controll()`, and `Iunctus::proprioception()`.

4.17.3 Member Data Documentation

4.17.3.1 `float* Controller::input` `[protected]`

Definition at line 27 of file `Controller.h`.

Referenced by `Controller()`.

4.17.3.2 `unsigned int Controller::nInputs`

Definition at line 23 of file `Controller.h`.

4.17.3.3 `unsigned int Controller::nOutputs`

Definition at line 24 of file `Controller.h`.

4.17.3.4 `float* Controller::output` `[protected]`

Definition at line 28 of file `Controller.h`.

Referenced by `Controller()`.

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

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

4.18 ElasticLink Class Reference

```
#include <ElasticLink.h>
```

Public Member Functions

- [ElasticLink](#) ([PhysicalObject](#) **object1*, [PhysicalObject](#) **object2*, *real* *length*, *real* *k*)
- virtual [~ElasticLink](#) ()
- void [applyForces](#) ()
- void [draw](#) ([GUI](#) **gui*)

Public Attributes

- [PhysicalObject](#) * *object1*
- [PhysicalObject](#) * *object2*
- *real* *length*
- *real* *k*

4.18.1 Detailed Description

An elastic link, linking two objects. The objects are connected in their reference point.

Definition at line 12 of file ElasticLink.h.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 [ElasticLink::ElasticLink](#) ([PhysicalObject](#) * *object1*, [PhysicalObject](#) * *object2*, *real* *length*, *real* *k*)

Definition at line 11 of file ElasticLink.cpp.

4.18.2.2 [ElasticLink::~~ElasticLink](#) () [virtual]

Definition at line 16 of file ElasticLink.cpp.

4.18.3 Member Function Documentation

4.18.3.1 void [ElasticLink::applyForces](#) ()

Applies the elastic forces to the objects, by adding them to the externalForces accumulator of the objects.

Definition at line 20 of file ElasticLink.cpp.

References [PhysicalObject::externalForce](#), [Vector2::getModule\(\)](#), *k*, *length*, *object1*, *object2*, [PhysicalObject::r](#), and *real*.

4.18.3.2 void ElasticLink::draw ([GUI * gui](#))

Definition at line 29 of file ElasticLink.cpp.

References [GUI::drawLine\(\)](#), [object1](#), [object2](#), and [PhysicalObject::r](#).

4.18.4 Member Data Documentation

4.18.4.1 [real ElasticLink::k](#)

The elastic contact of the link.

Definition at line 25 of file ElasticLink.h.

Referenced by [applyForces\(\)](#), and [Pac::Pac\(\)](#).

4.18.4.2 [real ElasticLink::length](#)

The rest length of the elastic link.

Definition at line 22 of file ElasticLink.h.

Referenced by [applyForces\(\)](#), and [Pac::controll\(\)](#).

4.18.4.3 [PhysicalObject* ElasticLink::object1](#)

Definition at line 18 of file ElasticLink.h.

Referenced by [applyForces\(\)](#), and [draw\(\)](#).

4.18.4.4 [PhysicalObject* ElasticLink::object2](#)

Definition at line 19 of file ElasticLink.h.

Referenced by [applyForces\(\)](#), and [draw\(\)](#).

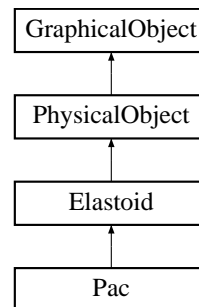
The documentation for this class was generated from the following files:

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

4.19 Elastoid Class Reference

```
#include <Elastoid.h>
```

Inheritance diagram for Elastoid::



Public Member Functions

- [Elastoid \(\)](#)
- virtual [~Elastoid \(\)](#)
- void [addObject \(PhysicalObject *object\)](#)
- virtual void [registerPrimitives \(Simulator *simulator\)](#)
- virtual void [computeDerivativesWithoutContacts \(ContactSolver *contactSolver\)](#)
- virtual void [computeDerivatives \(GlobalContactInfoVector *globalContacts\)](#)
- virtual void [integrate \(const Integrator &integrator\)](#)
- virtual bool [detectContacts \(PhysicalObject *object, GlobalContactInfoVector *contacts\)](#)
- virtual void [detectInternalContacts \(GlobalContactInfoVector *globalContacts\)](#)
- virtual void [deleteContacts \(\)](#)
- virtual bool [detectMouseContact \(const Vector2 &rMouse, Vector2 &p, PhysicalObject *&object\)](#)
- virtual void [draw \(GUI *gui\)](#)

Public Attributes

- [PhysicalObjectPVector objects](#)
- [ElasticLinkVector links](#)

4.19.1 Constructor & Destructor Documentation

4.19.1.1 Elastoid::Elastoid ()

Definition at line 13 of file Elastoid.cpp.

4.19.1.2 Elastoid::~~Elastoid () [virtual]

Definition at line 16 of file Elastoid.cpp.

References [objects](#), and [purgeContainer\(\)](#).

4.19.2 Member Function Documentation

4.19.2.1 void Elastoid::addObject ([PhysicalObject](#) * *object*) [inline]

Definition at line 18 of file Elastoid.h.

References objects.

Referenced by Pac::Pac().

4.19.2.2 void Elastoid::computeDerivatives ([GlobalContactInfoVector](#) * *globalContacts*) [virtual]

Computes the derivatives, given the contact forces. The globalContacts pointer is needed for contact processing for articulated agents.

Reimplemented from [PhysicalObject](#).

Definition at line 35 of file Elastoid.cpp.

References objects.

4.19.2.3 void Elastoid::computeDerivativesWithoutContacts ([ContactSolver](#) * *contactSolver*) [virtual]

Computes the derivatives (velocity, angular velocity) of the object, ignoring any eventual contact forces.

Reimplemented from [PhysicalObject](#).

Definition at line 27 of file Elastoid.cpp.

References links, and objects.

4.19.2.4 void Elastoid::deleteContacts () [virtual]

Delete the contacts of the object.

Reimplemented from [PhysicalObject](#).

Definition at line 67 of file Elastoid.cpp.

References objects.

4.19.2.5 bool Elastoid::detectContacts ([PhysicalObject](#) * *object*, [GlobalContactInfoVector](#) * *contacts*) [virtual]

Contact handling.

Implements [PhysicalObject](#).

Definition at line 52 of file Elastoid.cpp.

References objects.

4.19.2.6 void Elastoid::detectInternalContacts ([GlobalContactInfoVector](#) * *globalContacts*) [virtual]

Detects internal contacts. Does nothing for primitive objects or composed objects. To be overridden by complex objects that have parts that move relative to each other, such as articulated objects.

Reimplemented from [PhysicalObject](#).

Definition at line 60 of file Elastoid.cpp.

References objects.

4.19.2.7 bool Elastoid::detectMouseContact (const [Vector2](#) & *rMouse*, [Vector2](#) & *p*, [PhysicalObject](#) *& *object*) [virtual]

Detects whether a click of the mouse at *rMouse* has touched the object. In case of contact, returns true and sets *p* to the relative vector between the center of the object and the contact point; *p* is expressed in the object reference frame because it rotates with the object while the object is dragged. The object needs to be set by the function because, for articulated objects, the dragged object is an articulation and not the whole articulated object. To be overridden by the various object primitives.

Reimplemented from [PhysicalObject](#).

Definition at line 73 of file Elastoid.cpp.

References objects.

4.19.2.8 void Elastoid::draw ([GUI](#) * *gui*) [virtual]

Draw the object to the graphical user interface.

Implements [GraphicalObject](#).

Definition at line 84 of file Elastoid.cpp.

References links, objects, and GUI::setPenColor().

4.19.2.9 void Elastoid::integrate (const [Integrator](#) & *integrator*) [virtual]

Advances the time to the next timestep.

Reimplemented from [PhysicalObject](#).

Definition at line 41 of file Elastoid.cpp.

References objects, [Vector2::setXY\(\)](#), [Vector2::updateMax\(\)](#), and [Vector2::updateMin\(\)](#).

4.19.2.10 void Elastoid::registerPrimitives ([Simulator](#) * *simulator*) [virtual]

Registers all composing primitives of the object with the simulator.

Reimplemented from [PhysicalObject](#).

Definition at line 20 of file Elastoid.cpp.

References objects, and [Simulator::registerPrimitive\(\)](#).

4.19.3 Member Data Documentation

4.19.3.1 [ElasticLinkVector Elastoid::links](#)

Definition at line 16 of file Elastoid.h.

Referenced by `computeDerivativesWithoutContacts()`, and `draw()`.

4.19.3.2 [PhysicalObjectPVector Elastoid::objects](#)

Definition at line 15 of file Elastoid.h.

Referenced by `addObject()`, `computeDerivatives()`, `computeDerivativesWithoutContacts()`, `deleteContacts()`, `detectContacts()`, `detectInternalContacts()`, `detectMouseContact()`, `draw()`, `integrate()`, `registerPrimitives()`, and `~Elastoid()`.

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

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

4.20 GlobalContactInfo Class Reference

```
#include <GlobalContactInfo.h>
```

Public Member Functions

- [GlobalContactInfo](#) ([ContactInfo](#) *contact1, [ContactInfo](#) *contact2, [real](#) penetration=0.0)

Public Attributes

- [ContactInfo](#) * contact1
- [ContactInfo](#) * contact2
- [real](#) force
- [real](#) penetration
- [int](#) alpha

4.20.1 Constructor & Destructor Documentation

4.20.1.1 [GlobalContactInfo::GlobalContactInfo](#) ([ContactInfo](#) * contact1, [ContactInfo](#) * contact2, [real](#) penetration = 0.0) [inline]

Definition at line 13 of file GlobalContactInfo.h.

4.20.2 Member Data Documentation

4.20.2.1 [int](#) [GlobalContactInfo::alpha](#)

The index of this contact in the global contact vector.

Definition at line 31 of file GlobalContactInfo.h.

4.20.2.2 [ContactInfo*](#) [GlobalContactInfo::contact1](#)

Pointers to the contacts of the two objects that compose the contact. The GlobalContactInfo does not own the [ContactInfo](#) objects, should NOT delete them on destruction.

Definition at line 24 of file GlobalContactInfo.h.

4.20.2.3 [ContactInfo*](#) [GlobalContactInfo::contact2](#)

Definition at line 25 of file GlobalContactInfo.h.

4.20.2.4 [real](#) [GlobalContactInfo::force](#)

The modulus of the contact force.

Definition at line 27 of file GlobalContactInfo.h.

4.20.2.5 [real GlobalContactInfo::penetration](#)

The penetration area.

Definition at line 29 of file GlobalContactInfo.h.

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

- [GlobalContactInfo.h](#)

4.21 Graph Class Reference

```
#include <Graph.h>
```

Public Member Functions

- **Graph** (int *width*, int *height*, int *bufferSize*, int *nSubGraphs*=1, float *min*=0, float *max*=1)
- virtual **~Graph** ()
- void **push** (float value, int *nSubGraph*=0)
- void **drawBuffer** ()
- void **draw** (wxDC *outdc, int x, int y)

Public Attributes

- int *width*
- int *height*
- float *min*
- float *max*
- int *bufferSize*
- **GraphData** ** *subGraphs*
- int *nSubGraphs*
- wxBitmap *bitmap*
- wxMemoryDC *dc*

4.21.1 Detailed Description

This class manages the drawing of a graphic of the evolution in time of one or more quantities.

Definition at line 8 of file Graph.h.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 **Graph::Graph** (int *width*, int *height*, int *bufferSize*, int *nSubGraphs* = 1, float *min* = 0, float *max* = 1)

Definition at line 11 of file Graph.cpp.

References *bitmap*, *dc*, and *subGraphs*.

4.21.2.2 **Graph::~~Graph** () [virtual]

Definition at line 24 of file Graph.cpp.

References *subGraphs*.

4.21.3 Member Function Documentation

4.21.3.1 void Graph::draw (wxDC * *outdc*, int *x*, int *y*)

Definition at line 76 of file Graph.cpp.

References `dc`, `drawBuffer()`, `height`, and `width`.

4.21.3.2 void Graph::drawBuffer ()

Definition at line 37 of file Graph.cpp.

References `GraphData::dataSize`, `dc`, `GraphData::getValue()`, `height`, `max`, `min`, `subGraphs`, and `width`.

Referenced by `draw()`.

4.21.3.3 void Graph::push (float *value*, int *nSubGraph* = 0)

Adds a data value to the specified subgraph.

Definition at line 31 of file Graph.cpp.

References `max`, `min`, `GraphData::push()`, and `subGraphs`.

4.21.4 Member Data Documentation

4.21.4.1 wxBitmap Graph::bitmap

Bitmap used for internal buffer.

Definition at line 28 of file Graph.h.

Referenced by `Graph()`.

4.21.4.2 int Graph::bufferSize

The size of the memory buffer of the graph

Definition at line 18 of file Graph.h.

4.21.4.3 wxMemoryDC Graph::dc

DC used for internal buffer.

Definition at line 30 of file Graph.h.

Referenced by `draw()`, `drawBuffer()`, and `Graph()`.

4.21.4.4 int Graph::height

The dimensions, in pixels, of the graphical area on which the graph is drawn.

Definition at line 14 of file Graph.h.

Referenced by `draw()`, and `drawBuffer()`.

4.21.4.5 float [Graph::max](#)

Definition at line 15 of file Graph.h.

Referenced by `drawBuffer()`, and `push()`.

4.21.4.6 float [Graph::min](#)

Definition at line 15 of file Graph.h.

Referenced by `drawBuffer()`, and `push()`.

4.21.4.7 int [Graph::nSubGraphs](#)

Definition at line 22 of file Graph.h.

4.21.4.8 [GraphData Graph::subGraphs](#)**

The composing subgraph. More than one subgraph (curves) can be displayed in the same graph.

Definition at line 21 of file Graph.h.

Referenced by `drawBuffer()`, `Graph()`, `push()`, and `~Graph()`.

4.21.4.9 int [Graph::width](#)

The dimensions, in pixels, of the graphical area on which the graph is drawn.

Definition at line 14 of file Graph.h.

Referenced by `draw()`, and `drawBuffer()`.

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

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

4.22 GraphData Class Reference

```
#include <GraphData.h>
```

Public Member Functions

- [GraphData](#) (int [bufferSize](#))
- virtual [~GraphData](#) ()
- void [push](#) (float value)
- float [getValue](#) (int i)

Public Attributes

- int [bufferSize](#)
- int [dataSize](#)
- int [index](#)

Private Attributes

- float * [data](#)

4.22.1 Constructor & Destructor Documentation

4.22.1.1 [GraphData::GraphData](#) (int *bufferSize*)

Definition at line 3 of file [GraphData.cpp](#).

References [data](#), [dataSize](#), and [index](#).

4.22.1.2 [GraphData::~~GraphData](#) () [virtual]

Definition at line 10 of file [GraphData.cpp](#).

4.22.2 Member Function Documentation

4.22.2.1 float [GraphData::getValue](#) (int *i*) [inline]

Definition at line 33 of file [GraphData.h](#).

Referenced by [Graph::drawBuffer](#)().

4.22.2.2 void [GraphData::push](#) (float *value*) [inline]

Definition at line 23 of file [GraphData.h](#).

Referenced by [Graph::push](#)().

4.22.3 Member Data Documentation

4.22.3.1 `int GraphData::bufferSize`

The size of the memory buffer of the graph

Definition at line 12 of file GraphData.h.

4.22.3.2 `float* GraphData::data` `[private]`

Definition at line 39 of file GraphData.h.

Referenced by GraphData().

4.22.3.3 `int GraphData::dataSize`

The actual size of the data that was already set.

Definition at line 15 of file GraphData.h.

Referenced by Graph::drawBuffer(), and GraphData().

4.22.3.4 `int GraphData::index`

The index that marks the start of the data. Virtually, the new data values added to the graph are pushed at the end of the data stack, while old values are popped at the beginning. In practice, just the index that marks the start of the data moves, while the new data values are overwritten over the older ones.

Definition at line 21 of file GraphData.h.

Referenced by GraphData().

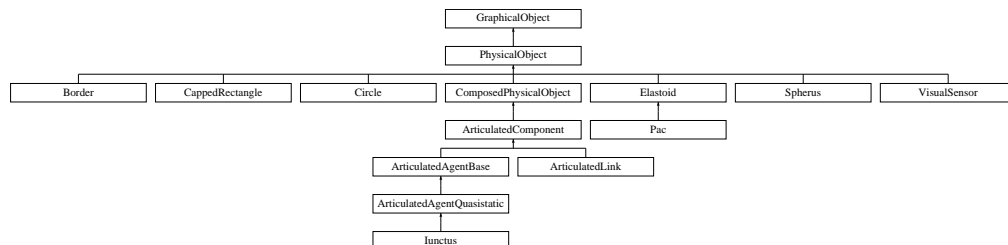
The documentation for this class was generated from the following files:

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

4.23 GraphicalObject Class Reference

```
#include <GraphicalObject.h>
```

Inheritance diagram for GraphicalObject::



Public Member Functions

- [GraphicalObject](#) ([Color](#) outlineColor=[GUI::colorBlack](#), [Color](#) fillColor=[GUI::colorTransparent](#))
- virtual [~GraphicalObject](#) ()
- virtual void [draw](#) ([GUI](#) *gui)=0
- virtual void [setOutlineColor](#) ([Color](#) color)
- virtual void [setFillColor](#) ([Color](#) color)
- void [setColor](#) ([Color](#) color)

Public Attributes

- [Color](#) outlineColor
- [Color](#) fillColor

4.23.1 Detailed Description

Base class for all objects that can be drawn (including physical objects, but eventually also elements like markers, contacts, forces, etc.). It is characterized by an outline color and a fill color.

Definition at line 9 of file [GraphicalObject.h](#).

4.23.2 Constructor & Destructor Documentation

4.23.2.1 [GraphicalObject::GraphicalObject](#) ([Color](#) outlineColor = [GUI::colorBlack](#), [Color](#) fillColor = [GUI::colorTransparent](#))

Definition at line 3 of file [GraphicalObject.cpp](#).

4.23.2.2 [GraphicalObject::~~GraphicalObject](#) () [virtual]

Destructor, does nothing.

Definition at line 8 of file [GraphicalObject.cpp](#).

4.23.3 Member Function Documentation

4.23.3.1 **virtual void GraphicalObject::draw** (**GUI** * *gui*) [pure virtual]

Draw the object to the graphical user interface.

Implemented in [Iunctus](#), [Spherus](#), [Border](#), [CappedRectangle](#), [Circle](#), [ComposedPhysicalObject](#), [Elastoid](#), [VisualSensor](#), [ArticulatedAgentBase](#), and [ArticulatedLink](#).

Referenced by [ComposedPhysicalObject::draw\(\)](#).

4.23.3.2 **void GraphicalObject::setColor** (**Color** *color*) [inline]

Sets both outline and fill color to the same color.

Definition at line 26 of file [GraphicalObject.h](#).

Referenced by [Pac::Pac\(\)](#).

4.23.3.3 **virtual void GraphicalObject::setFillColor** (**Color** *color*) [inline, virtual]

Sets fill color.

Reimplemented in [ComposedPhysicalObject](#), and [ArticulatedAgentBase](#).

Definition at line 23 of file [GraphicalObject.h](#).

4.23.3.4 **virtual void GraphicalObject::setOutlineColor** (**Color** *color*) [inline, virtual]

Sets outline color.

Reimplemented in [ComposedPhysicalObject](#), and [ArticulatedAgentBase](#).

Definition at line 20 of file [GraphicalObject.h](#).

4.23.4 Member Data Documentation

4.23.4.1 **Color GraphicalObject::fillColor**

The fill color of the object.

Definition at line 35 of file [GraphicalObject.h](#).

4.23.4.2 **Color GraphicalObject::outlineColor**

The outline color of the object.

Definition at line 32 of file [GraphicalObject.h](#).

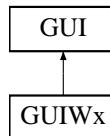
The documentation for this class was generated from the following files:

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

4.24 GUI Class Reference

```
#include <GUI.h>
```

Inheritance diagram for GUI::



Public Member Functions

- **GUI** (float iniPanX=0, float iniPanY=400, float zoom=100.0f, int signX=1, int signY=-1)
- virtual **~GUI** ()
- void **setZoom** (float newZoom)
- virtual void **clear** ()=0
- virtual void **drawCircle** (float x, float y, float r)=0
- void **drawCircle** (const **Vector2** &r, float R)
- virtual void **drawLine** (float x1, float y1, float x2, float y2)=0
- void **drawLine** (const **Vector2** &r1, const **Vector2** &r2)
- void **drawLine** (const **Vector2** &r1, float x2, float y2)
- virtual void **drawRectangle** (float x, float y, float w, float h)=0
- virtual void **drawCappedRectangle** (float x, float y, float l, float R, float alpha)=0
- virtual void **setBrushColor** (**Color** color)=0
- virtual void **setPenColor** (**Color** color)=0
- virtual void **drawArrow** (float x1, float x2, float y1, float y2)=0
- virtual void **outText** (const char *s)=0
- virtual void **outTextStatusBar** (const char *s, int pos)=0
- virtual void **drawForce** (const **Vector2** &r, const **Vector2** &force)=0
- virtual void **drawTorque** (const **Vector2** &r, const **real** alpha, const **real** torque)=0
- float **mapX** (float x)
- float **mapY** (float y)
- float **inverseMapX** (float x)
- float **inverseMapY** (float y)
- float **mapLen** (float len)

Public Attributes

- float **panX**
- float **panY**
- float **zoom**
- int **signX**
- int **signY**
- float **zoomX**
- float **zoomY**

Static Public Attributes

- const [Color](#) colorTransparent = [Color](#)(0,0,0,true)
- const [Color](#) colorWhite = [Color](#)(255,255,255)
- const [Color](#) colorBlack = [Color](#)(0,0,0)

4.24.1 Constructor & Destructor Documentation

4.24.1.1 [GUI::GUI](#) (float *iniPanX* = 0, float *iniPanY* = 400, float *zoom* = 100.0f, int *signX* = 1, int *signY* = -1)

Definition at line 16 of file [GUI.cpp](#).

4.24.1.2 [GUI::~~GUI](#) () [virtual]

Definition at line 27 of file [GUI.cpp](#).

4.24.2 Member Function Documentation

4.24.2.1 virtual void [GUI::clear](#) () [pure virtual]

Erases everything.

Implemented in [GUIWx](#).

4.24.2.2 virtual void [GUI::drawArrow](#) (float *x1*, float *x2*, float *y1*, float *y2*) [pure virtual]

Implemented in [GUIWx](#).

4.24.2.3 virtual void [GUI::drawCappedRectangle](#) (float *x*, float *y*, float *l*, float *R*, float *alpha*) [pure virtual]

Implemented in [GUIWx](#).

Referenced by [CappedRectangle::draw](#)().

4.24.2.4 void [GUI::drawCircle](#) (const [Vector2](#) &*r*, float *R*) [inline]

Definition at line 29 of file [GUI.h](#).

References [Vector2::x](#), and [Vector2::y](#).

4.24.2.5 virtual void [GUI::drawCircle](#) (float *x*, float *y*, float *r*) [pure virtual]

Implemented in [GUIWx](#).

Referenced by [Circle::draw](#)(), and [ArticulatedLink::draw](#)().

4.24.2.6 `virtual void GUI::drawForce (const Vector2 & r, const Vector2 & force)` [pure virtual]

Draws a force vector, having r as origin and force as magnitude.

Implemented in [GUIWx](#).

Referenced by Spherus::draw(), and PhysicalObject::drawContactForces().

4.24.2.7 `void GUI::drawLine (const Vector2 & r1, float x2, float y2)` [inline]

Definition at line 35 of file GUI.h.

References Vector2::x, and Vector2::y.

4.24.2.8 `void GUI::drawLine (const Vector2 & r1, const Vector2 & r2)` [inline]

Definition at line 32 of file GUI.h.

References Vector2::x, and Vector2::y.

4.24.2.9 `virtual void GUI::drawLine (float x1, float y1, float x2, float y2)` [pure virtual]

Implemented in [GUIWx](#).

Referenced by VisualSensor::draw(), Spherus::draw(), Iunctus::draw(), ElasticLink::draw(), Circle::drawSensors(), and CappedRectangle::drawSensors().

4.24.2.10 `virtual void GUI::drawRectangle (float x, float y, float w, float h)` [pure virtual]

Draws a rectangle with the given bottom left corner, and with the given size.

Implemented in [GUIWx](#).

4.24.2.11 `virtual void GUI::drawTorque (const Vector2 & r, const real alpha, const real torque)` [pure virtual]

Draws a torque, having r as rotation point, alpha as starting angle and torque as magnitude.

Implemented in [GUIWx](#).

4.24.2.12 `float GUI::inverseMapX (float x)` [inline]

Definition at line 70 of file GUI.h.

Referenced by ThyrixMainFrame::onLeftDown(), ThyrixMainFrame::onLeftUp(), ThyrixMainFrame::onMotion(), ThyrixMainFrame::onRightDown(), and ThyrixMainFrame::onRightUp().

4.24.2.13 `float GUI::inverseMapY (float y)` [inline]

Definition at line 74 of file GUI.h.

Referenced by ThyrixMainFrame::onLeftDown(), ThyrixMainFrame::onLeftUp(), ThyrixMainFrame::onMotion(), ThyrixMainFrame::onRightDown(), and ThyrixMainFrame::onRightUp().

4.24.2.14 float GUI::mapLen (float *len*) [inline]

Definition at line 78 of file GUI.h.

Referenced by GUIWx::drawCircle().

4.24.2.15 float GUI::mapX (float *x*) [inline]

Definition at line 60 of file GUI.h.

Referenced by GUIWx::drawCappedRectangle(), GUIWx::drawCircle(), GUIWx::drawLine(), GUIWx::drawRectangle(), and GUIWx::drawTorque().

4.24.2.16 float GUI::mapY (float *y*) [inline]

Definition at line 65 of file GUI.h.

Referenced by GUIWx::drawCappedRectangle(), GUIWx::drawCircle(), GUIWx::drawLine(), GUIWx::drawRectangle(), and GUIWx::drawTorque().

4.24.2.17 virtual void GUI::outText (const char * *s*) [pure virtual]

Outputs a text string.

Implemented in [GUIWx](#).

4.24.2.18 virtual void GUI::outTextStatusBar (const char * *s*, int *pos*) [pure virtual]

Outputs a text to the statusbar, at position pos.

Implemented in [GUIWx](#).

Referenced by World::draw().

4.24.2.19 virtual void GUI::setBrushColor ([Color](#) *color*) [pure virtual]

Implemented in [GUIWx](#).

Referenced by Spherus::draw(), Simulator::draw(), Circle::draw(), and CappedRectangle::draw().

4.24.2.20 virtual void GUI::setPenColor ([Color](#) *color*) [pure virtual]

Implemented in [GUIWx](#).

Referenced by VisualSensor::draw(), Spherus::draw(), Elastoid::draw(), Circle::draw(), CappedRectangle::draw(), and ArticulatedLink::draw().

4.24.2.21 void GUI::setZoom (float *newZoom*)

Definition at line 30 of file GUI.cpp.

References zoom, zoomX, and zoomY.

4.24.3 Member Data Documentation

4.24.3.1 `const Color GUI::colorBlack = Color(0,0,0) [static]`

Definition at line 13 of file GUI.cpp.

4.24.3.2 `const Color GUI::colorTransparent = Color(0,0,0,true) [static]`

Definition at line 11 of file GUI.cpp.

4.24.3.3 `const Color GUI::colorWhite = Color(255,255,255) [static]`

Definition at line 12 of file GUI.cpp.

4.24.3.4 `float GUI::panX`

Definition at line 83 of file GUI.h.

4.24.3.5 `float GUI::panY`

Definition at line 84 of file GUI.h.

4.24.3.6 `int GUI::signX`

Definition at line 86 of file GUI.h.

4.24.3.7 `int GUI::signY`

Definition at line 87 of file GUI.h.

4.24.3.8 `float GUI::zoom`

Definition at line 85 of file GUI.h.

Referenced by setZoom().

4.24.3.9 `float GUI::zoomX`

Definition at line 88 of file GUI.h.

Referenced by setZoom().

4.24.3.10 `float GUI::zoomY`

Definition at line 89 of file GUI.h.

Referenced by setZoom().

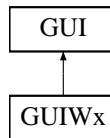
The documentation for this class was generated from the following files:

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

4.25 GUIWx Class Reference

```
#include <GUIWx.h>
```

Inheritance diagram for GUIWx::



Public Member Functions

- **GUIWx** (int iniPanX, int iniPanY, float **zoom**, wxStatusBar *iniStatusBar)
- virtual **~GUIWx** ()
- void **setDC** (wxDC *iniDC)
- virtual void **clear** ()
- void **drawCircle** (float x, float y, float r)
- void **drawLine** (float x1, float y1, float x2, float y2)
- void **drawRectangle** (float x, float y, float w, float h)
- void **drawCappedRectangle** (float x, float y, float l, float R, float alpha)
- void **setBrushColor** (Color color)
- void **setPenColor** (Color color)
- virtual void **drawArrow** (float x1, float x2, float y1, float y2)
- void **drawForce** (const **Vector2** &r, const **Vector2** &force)
- void **drawTorque** (const **Vector2** &r, const **real** alpha, const **real** torque)
- void **outText** (const char *s)
- void **outTextStatusBar** (const char *s, int pos=0)

Public Attributes

- wxDC * **dc**

Private Attributes

- wxStatusBar * **statusBar**

4.25.1 Constructor & Destructor Documentation

4.25.1.1 GUIWx::GUIWx (int *iniPanX*, int *iniPanY*, float *zoom*, wxStatusBar * *iniStatusBar*)

Parameters: the width and height of the window, zoom applied to simulator values.

Definition at line 8 of file GUIWx.cpp.

4.25.1.2 GUIWx::~GUIWx () [virtual]

Definition at line 14 of file GUIWx.cpp.

4.25.2 Member Function Documentation

4.25.2.1 void GUIWx::clear () [virtual]

Erases everything.

Implements [GUI](#).

Definition at line 21 of file GUIWx.cpp.

References [dc](#).

4.25.2.2 void GUIWx::drawArrow (float x1, float y1, float x2, float y2) [virtual]

draws an arrow from x1,y1 to x2,y2

Implements [GUI](#).

Definition at line 116 of file GUIWx.cpp.

References [drawLine\(\)](#), and [M_PI](#).

Referenced by [drawForce\(\)](#).

4.25.2.3 void GUIWx::drawCappedRectangle (float x, float y, float l, float R, float alpha) [virtual]

Implements [GUI](#).

Definition at line 45 of file GUIWx.cpp.

References [cosSum\(\)](#), [dc](#), [M_PI](#), [GUI::mapX\(\)](#), [GUI::mapY\(\)](#), and [sinSum\(\)](#).

4.25.2.4 void GUIWx::drawCircle (float x, float y, float r) [virtual]

Draws a circle.

Implements [GUI](#).

Definition at line 25 of file GUIWx.cpp.

References [dc](#), [GUI::mapLen\(\)](#), [GUI::mapX\(\)](#), and [GUI::mapY\(\)](#).

4.25.2.5 void GUIWx::drawForce (const [Vector2](#) & r, const [Vector2](#) & force) [virtual]

Draws a force vector, having r as origin and force as magnitude.

Implements [GUI](#).

Definition at line 150 of file GUIWx.cpp.

References [drawArrow\(\)](#), [real](#), [Vector2::x](#), and [Vector2::y](#).

4.25.2.6 void GUIWx::drawLine (float x1, float y1, float x2, float y2) [virtual]

Draws a line

Implements [GUI](#).

Definition at line 29 of file GUIWx.cpp.

References `dc`, `GUI::mapX()`, and `GUI::mapY()`.

Referenced by `drawArrow()`, and `drawTorque()`.

4.25.2.7 void GUIWx::drawRectangle (float *x*, float *y*, float *w*, float *h*) [virtual]

Draws a rectangle with the given bottom left corner, and with the given size

Implements [GUI](#).

Definition at line 33 of file GUIWx.cpp.

References `dc`, `GUI::mapX()`, and `GUI::mapY()`.

4.25.2.8 void GUIWx::drawTorque (const [Vector2](#) & *r*, const [real](#) *alpha*, const [real](#) *torque*) [virtual]

Draws a torque, having *r* as rotation point, *alpha* as starting angle and *torque* as magnitude.

Implements [GUI](#).

Definition at line 158 of file GUIWx.cpp.

References `dc`, `drawLine()`, `M_PI`, `GUI::mapX()`, `GUI::mapY()`, `real`, `Vector2::x`, and `Vector2::y`.

4.25.2.9 void GUIWx::outText (const char * *s*) [virtual]

Outputs a text string.

Implements [GUI](#).

Definition at line 105 of file GUIWx.cpp.

References `outTextStatusBar()`.

4.25.2.10 void GUIWx::outTextStatusBar (const char * *s*, int *pos* = 0) [virtual]

Outputs a text to the statusbar, at position *pos*.

Implements [GUI](#).

Definition at line 109 of file GUIWx.cpp.

References `statusBar`.

Referenced by `outText()`.

4.25.2.11 void GUIWx::setBrushColor ([Color](#) *color*) [virtual]

Implements [GUI](#).

Definition at line 89 of file GUIWx.cpp.

References `Color::b`, `dc`, `Color::g`, `Color::r`, and `Color::transparent`.

4.25.2.12 void GUIWx::setDC (wxDC * *iniDC*)

Definition at line 17 of file GUIWx.cpp.

References [dc](#).

Referenced by [ThyrixMainFrame::paintDC\(\)](#).

4.25.2.13 void GUIWx::setPenColor ([Color](#) *color*) [virtual]

Implements [GUI](#).

Definition at line 97 of file GUIWx.cpp.

References [Color::b](#), [dc](#), [Color::g](#), [Color::r](#), and [Color::transparent](#).

4.25.3 Member Data Documentation**4.25.3.1 wxDC* [GUIWx::dc](#)**

The device context.

Definition at line 55 of file GUIWx.h.

Referenced by [clear\(\)](#), [drawCappedRectangle\(\)](#), [drawCircle\(\)](#), [drawLine\(\)](#), [drawRectangle\(\)](#), [drawTorque\(\)](#), [setBrushColor\(\)](#), [setDC\(\)](#), and [setPenColor\(\)](#).

4.25.3.2 wxStatusBar* [GUIWx::statusBar](#) [private]

The status bar.

Definition at line 60 of file GUIWx.h.

Referenced by [outTextStatusBar\(\)](#).

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

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

4.26 Integrator Class Reference

```
#include <Integrator.h>
```

Public Member Functions

- [Integrator](#) ([real](#) iniDt)
- void [integrate](#) ([real](#) &x, const [real](#) &v) const
- void [integrate](#) ([Vector2](#) &x, const [Vector2](#) &v) const
- [real](#) [getDt](#) () const
- void [setDt](#) ([real](#) t)

Private Attributes

- [real](#) dt

4.26.1 Constructor & Destructor Documentation

4.26.1.1 [Integrator::Integrator](#) ([real](#) iniDt) [inline]

Definition at line 8 of file Integrator.h.

4.26.2 Member Function Documentation

4.26.2.1 [real](#) [Integrator::getDt](#) () const [inline]

Definition at line 20 of file Integrator.h.

References [real](#).

Referenced by [Simulator::advanceTime](#)(), and [Simulator::getDt](#)().

4.26.2.2 void [Integrator::integrate](#) ([Vector2](#) &x, const [Vector2](#) &v) const [inline]

Definition at line 15 of file Integrator.h.

References [Vector2::x](#), and [Vector2::y](#).

4.26.2.3 void [Integrator::integrate](#) ([real](#) &x, const [real](#) &v) const [inline]

Gets the value of x at the next timestep, given its derivative v.

Definition at line 11 of file Integrator.h.

Referenced by [PhysicalObject::integrate](#)(), and [ArticulatedLink::integrate](#)().

4.26.2.4 void [Integrator::setDt](#) ([real](#) t) [inline]

Definition at line 24 of file Integrator.h.

4.26.3 Member Data Documentation

4.26.3.1 `real Integrator::dt` [private]

Definition at line 29 of file Integrator.h.

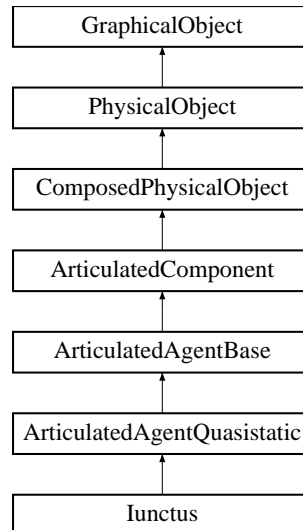
The documentation for this class was generated from the following file:

- [Integrator.h](#)

4.27 Iunctus Class Reference

```
#include <Iunctus.h>
```

Inheritance diagram for Iunctus::



Public Member Functions

- `Iunctus` (`real` x=2.0, `real` y=0.8, `std::string` label="Iunctus")
- virtual `~Iunctus` ()
- void `build` (`real` x=2.0, `real` y=0.8)
- void `proprioception` ()
- virtual void `deleteContacts` ()
- void `controll` ()
- void `draw` (`GUI` *gui)

Public Attributes

- `Circle` * myBody
- `VisualSensor` * eye
- `RandomController` * controller

4.27.1 Constructor & Destructor Documentation

4.27.1.1 `Iunctus::Iunctus` (`real` x = 2.0, `real` y = 0.8, `std::string` label = "Iunctus")

Definition at line 15 of file `Iunctus.cpp`.

References `build()`, `controller`, and `eye`.

4.27.1.2 `Iunctus::~~Iunctus` () [virtual]

Definition at line 22 of file `Iunctus.cpp`.

4.27.2 Member Function Documentation

4.27.2.1 void Iunctus::build (real $x = 2.0$, real $y = 0.8$)

Definition at line 27 of file Iunctus.cpp.

References [ArticulatedAgentBase::addLimb\(\)](#), [ComposedPhysicalObject::addObject\(\)](#), [ArticulatedLimb::addOneObjectLink\(\)](#), [ArticulatedComponent::computeIStar0\(\)](#), [ComposedPhysicalObject::computeMemberPositions\(\)](#), [degrees](#), [eye](#), [ArticulatedAgentQuasistatic::forwardKinematics\(\)](#), [ArticulatedLink::k](#), [M_PI](#), [myBody](#), [real](#), [ArticulatedAgentBase::setFillColor\(\)](#), [ArticulatedAgentBase::setOutlineColor\(\)](#), [Vector2::setXY\(\)](#), [ArticulatedLink::theta0](#), [ArticulatedLink::theta0Max](#), and [ArticulatedLink::theta0Min](#).

Referenced by [Iunctus\(\)](#).

4.27.2.2 void Iunctus::control () [virtual]

Transmits perceptual information to the controller and gets motor information from it. To be overridden by agents or other objects with self-generated movement.

Reimplemented from [PhysicalObject](#).

Definition at line 143 of file Iunctus.cpp.

References [RandomController::advanceTime\(\)](#), [controller](#), [Controller::getOutput\(\)](#), [ArticulatedLimb::links](#), [myBody](#), [Circle::R](#), [real](#), [ArticulatedLink::theta0](#), [ArticulatedLink::theta0Max](#), [ArticulatedLink::theta0Min](#), [Vector3::x](#), [Vector3::y](#), and [Vector3::z](#).

4.27.2.3 virtual void Iunctus::deleteContacts () [inline, virtual]

Delete the contacts of the object.

Reimplemented from [ArticulatedAgentBase](#).

Definition at line 26 of file Iunctus.h.

References [PhysicalObject::deleteContacts\(\)](#), [ArticulatedAgentBase::deleteContacts\(\)](#), and [eye](#).

4.27.2.4 void Iunctus::draw (GUI * gui) [virtual]

Calls the draw methods for the body and the limbs.

Reimplemented from [ArticulatedAgentBase](#).

Definition at line 191 of file Iunctus.cpp.

References [PhysicalObject::alpha](#), [ArticulatedAgentBase::draw\(\)](#), [GUI::drawLine\(\)](#), [M_PI](#), [myBody](#), [Circle::R](#), [PhysicalObject::r](#), [real](#), [Vector2::x](#), and [Vector2::y](#).

4.27.2.5 void Iunctus::proprioception ()

Definition at line 123 of file Iunctus.cpp.

References [controller](#), [ArticulatedLimb::links](#), [Controller::setInput\(\)](#), [ArticulatedLink::theta](#), [ArticulatedLink::theta0Max](#), and [ArticulatedLink::theta0Min](#).

4.27.3 Member Data Documentation

4.27.3.1 [RandomController*](#) [Iunctus::controller](#)

Definition at line 35 of file Iunctus.h.

Referenced by [controll\(\)](#), [Iunctus\(\)](#), and [proprioception\(\)](#).

4.27.3.2 [VisualSensor*](#) [Iunctus::eye](#)

Definition at line 20 of file Iunctus.h.

Referenced by [build\(\)](#), [deleteContacts\(\)](#), [IunctusSimulator::detectContacts\(\)](#), and [Iunctus\(\)](#).

4.27.3.3 [Circle*](#) [Iunctus::myBody](#)

Definition at line 19 of file Iunctus.h.

Referenced by [build\(\)](#), [controll\(\)](#), and [draw\(\)](#).

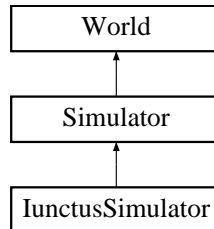
The documentation for this class was generated from the following files:

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

4.28 IunctusSimulator Class Reference

```
#include <IunctusSimulator.h>
```

Inheritance diagram for IunctusSimulator::



Public Member Functions

- [IunctusSimulator \(\)](#)
- virtual [~IunctusSimulator \(\)](#)
- void [init \(\)](#)
- virtual void [detectContacts \(\)](#)

Public Attributes

- [Iunctus * agent](#)

4.28.1 Detailed Description

A simulator that includes the [Iunctus](#) agent.

Definition at line 14 of file IunctusSimulator.h.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 IunctusSimulator::IunctusSimulator ()

r, x, y.

Definition at line 8 of file IunctusSimulator.cpp.

References [ComposedPhysicalObject::addObject\(\)](#), [agent](#), [degrees](#), [real](#), and [Simulator::registerObject\(\)](#).

4.28.2.2 IunctusSimulator::~IunctusSimulator () [virtual]

Definition at line 52 of file IunctusSimulator.cpp.

4.28.3 Member Function Documentation

4.28.3.1 void IunctusSimulator::detectContacts () [virtual]

Reimplemented from [Simulator](#).

Definition at line 55 of file IunctusSimulator.cpp.

References `agent`, `VisualSensor::detectContacts()`, `Simulator::detectContacts()`, and `Iunctus::eye`.

4.28.3.2 `void IunctusSimulator::init ()`

4.28.4 Member Data Documentation

4.28.4.1 `Iunctus*` `IunctusSimulator::agent`

Definition at line 21 of file IunctusSimulator.h.

Referenced by `detectContacts()`, and `IunctusSimulator()`.

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

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

4.29 LinkContactInfo Class Reference

```
#include <LinkContactInfo.h>
```

Public Member Functions

- [LinkContactInfo](#) ()
- [LinkContactInfo](#) (const int &[alpha](#), const [Vector3](#) &[v](#))
- virtual [~LinkContactInfo](#) ()
- bool [operator<](#) (const [LinkContactInfo](#) &[other](#)) const

Public Attributes

- [Vector3](#) [v](#)
- real [thetaFactor](#)
- int [alpha](#)

4.29.1 Constructor & Destructor Documentation

4.29.1.1 [LinkContactInfo::LinkContactInfo](#) () [inline]

Definition at line 14 of file [LinkContactInfo.h](#).

4.29.1.2 [LinkContactInfo::LinkContactInfo](#) (const int &*alpha*, const [Vector3](#) &*v*) [inline]

Definition at line 16 of file [LinkContactInfo.h](#).

4.29.1.3 virtual [LinkContactInfo::~LinkContactInfo](#) () [inline, virtual]

Definition at line 21 of file [LinkContactInfo.h](#).

4.29.2 Member Function Documentation

4.29.2.1 bool [LinkContactInfo::operator<](#) (const [LinkContactInfo](#) &*other*) const [inline]

Definition at line 27 of file [LinkContactInfo.h](#).

References [alpha](#).

4.29.3 Member Data Documentation

4.29.3.1 int [LinkContactInfo::alpha](#)

Definition at line 25 of file [LinkContactInfo.h](#).

Referenced by [operator<\(\)](#).

4.29.3.2 [real LinkContactInfo::thetaFactor](#)

Definition at line 24 of file LinkContactInfo.h.

4.29.3.3 [Vector3 LinkContactInfo::v](#)

Definition at line 23 of file LinkContactInfo.h.

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

- [LinkContactInfo.h](#)

4.30 MathTools Class Reference

```
#include <MathTools.h>
```

Public Member Functions

- [MathTools](#) ()
- virtual [~MathTools](#) ()

Static Public Member Functions

- template<class Number> Number [sqr](#) (Number x)
- template<class Number> int [sign](#) (Number x)
- template<class Number> double [setSign](#) (double x, Number sign)
- template<class Number> double [max](#) (Number a, Number b)
- double [modulus](#) (double x, double y)

Static Public Attributes

- const double [pi](#) = 3.1415926535897932384626433832795028841971693993751058209749445923078164062862
- const double [degrees](#) = [MathTools::pi](#)/180.0

4.30.1 Constructor & Destructor Documentation

4.30.1.1 [MathTools::MathTools](#) () [inline]

Definition at line 12 of file MathTools.h.

4.30.1.2 virtual [MathTools::~~MathTools](#) () [inline, virtual]

Definition at line 13 of file MathTools.h.

4.30.2 Member Function Documentation

4.30.2.1 template<class Number> double [MathTools::max](#) (Number *a*, Number *b*) [inline, static]

Definition at line 34 of file MathTools.h.

Referenced by [SystemSolver::svDecompose](#)().

4.30.2.2 double [MathTools::modulus](#) (double *x*, double *y*) [inline, static]

Returns $\sqrt{x^2+y^2}$ without destructive underflow or overflow. After Numerical Recipes, p. 70 (Press et al.)

Definition at line 40 of file MathTools.h.

References [sqr](#)().

Referenced by `VisualSensor::detectContacts()`, and `SystemSolver::svDecompose()`.

4.30.2.3 `template<class Number> double MathTools::setSign (double x, Number sign)`
[inline, static]

Returns a number having the absolute value of *x* and the sign *sign*.

Definition at line 30 of file `MathTools.h`.

Referenced by `SystemSolver::svDecompose()`.

4.30.2.4 `template<class Number> int MathTools::sign (Number x)` [inline, static]

Definition at line 25 of file `MathTools.h`.

4.30.2.5 `template<class Number> Number MathTools::sqr (Number x)` [inline, static]

Definition at line 21 of file `MathTools.h`.

4.30.3 Member Data Documentation

4.30.3.1 `const double MathTools::degrees = MathTools::pi/180.0` [static]

Defines the size of a degree in radians.

Definition at line 6 of file `MathTools.cpp`.

4.30.3.2 `const double MathTools::pi =`
`3.1415926535897932384626433832795028841971693993751058209749445923078164062862`
[static]

Defines pi (3.14...) in case that it is not defined by the C library used.

Definition at line 4 of file `MathTools.cpp`.

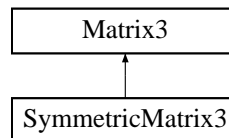
The documentation for this class was generated from the following files:

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

4.31 Matrix3 Class Reference

```
#include <Matrix3.h>
```

Inheritance diagram for Matrix3::



Public Member Functions

- [Matrix3](#) ()
- virtual [~Matrix3](#) ()
- void [setToZero](#) ()
- void [setElement](#) (int i, int j, [real](#) value)
- void [setRow](#) (int i, [real](#) v1, [real](#) v2, [real](#) v3)
- void [setColumn](#) (int i, const [Vector3](#) &v)
- [real](#) [getElement](#) (int i, int j)
- [real](#) [getDeterminant](#) ()
- [real](#) * [operator\[\]](#) (int i)
- [Matrix3](#) & [operator+=](#) (const [Matrix3](#) &m)

Public Attributes

- [real](#) [matrix](#) [3][3]

4.31.1 Detailed Description

A 3x3 matrix.

Definition at line 12 of file Matrix3.h.

4.31.2 Constructor & Destructor Documentation

4.31.2.1 [Matrix3::Matrix3](#) () [inline]

Default constructor, sets the matrix to 0.

Definition at line 16 of file Matrix3.h.

4.31.2.2 [Matrix3::~~Matrix3](#) () [virtual]

Definition at line 12 of file Matrix3.cpp.

4.31.3 Member Function Documentation

4.31.3.1 `real Matrix3::getDeterminant ()`

Gets the determinant of the matrix.

Definition at line 19 of file Matrix3.cpp.

References matrix, and real.

Referenced by ArticulatedAgentBase::solveSystem().

4.31.3.2 `real Matrix3::getElement (int i, int j)` `[inline]`

Gets element (*i*,*j*) of the matrix; *i*,*j* are numbered starting from 1 (*i*,*j*=1,2,3).

Definition at line 56 of file Matrix3.h.

References real.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), ArticulatedLink::computeForceQuasistatic(), and ArticulatedAgentQuasistatic::forwardAccelerations().

4.31.3.3 `Matrix3& Matrix3::operator+= (const Matrix3 & m)` `[inline]`

Addition to another matrix.

Definition at line 69 of file Matrix3.h.

References matrix.

4.31.3.4 `]`

`real * Matrix3::operator[] (int i)`

Gets row *i*; *i* is numbered starting from 0 (*i*=0,1,2).

Definition at line 46 of file Matrix3.cpp.

References matrix, and real.

4.31.3.5 `void Matrix3::setColumn (int i, const Vector3 & v)` `[inline]`

Sets column *i* of the matrix to *v*; *i* is numbered starting from 1 (*i*=1,2,3).

Definition at line 48 of file Matrix3.h.

References Vector3::x, Vector3::y, and Vector3::z.

Referenced by ArticulatedAgentBase::solveSystem().

4.31.3.6 `void Matrix3::setElement (int i, int j, real value)` `[inline]`

Sets element (*i*,*j*) to value *value*; *i*,*j* are numbered starting from 1 (*i*,*j*=1,2,3).

Definition at line 34 of file Matrix3.h.

4.31.3.7 void Matrix3::setRow (int *i*, [real v1](#), [real v2](#), [real v3](#)) [inline]

Sets row *i* of the matrix to (*v1*, *v2*, *v3*); *i* is numbered starting from 1 (*i*=1,2,3).

Definition at line 40 of file Matrix3.h.

Referenced by ArticulatedComponent::computeIStar0().

4.31.3.8 void Matrix3::setToZero () [inline]

Sets all matrix elements to 0.

Definition at line 27 of file Matrix3.h.

4.31.4 Member Data Documentation**4.31.4.1 [real Matrix3::matrix](#)[3][3]**

The placeholder for the data.

Definition at line 24 of file Matrix3.h.

Referenced by getDeterminant(), SymmetricMatrix3::operator+=((), operator+=((), and operator[]().

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

- [Matrix3.h](#)
- [Matrix3.cpp](#)

4.32 MTRandom Class Reference

```
#include <MTRandom.h>
```

Public Member Functions

- [MTRandom \(\)](#)
- virtual [~MTRandom \(\)](#)

Static Public Member Functions

- void [seed](#) (unsigned long)
- void [seed](#) (const unsigned long *, int size)
- unsigned long int [getLongInt](#) ()
- unsigned int [getInt](#) (unsigned int n)
- float [getFloat](#) ()
- double [getDouble](#) ()
- double [getDoubleClosed](#) ()
- double [getDoubleOpen](#) ()
- double [getDoubleHR](#) ()

Private Member Functions

- [MTRandom](#) (const [MTRandom](#) &)
- void [operator=](#) (const [MTRandom](#) &)

Static Private Member Functions

- void [gen_state](#) ()
- unsigned long [twiddle](#) (unsigned long u, unsigned long v)

Static Private Attributes

- unsigned long int [state](#) [[MTRandomN](#)]
- int [p](#) = 0

4.32.1 Constructor & Destructor Documentation

4.32.1.1 MTRandom::MTRandom ()

Definition at line 22 of file MTRandom.cpp.

4.32.1.2 MTRandom::~~MTRandom () [virtual]

Definition at line 25 of file MTRandom.cpp.

4.32.1.3 `MTRandom::MTRandom (const MTRandom &) [private]`

4.32.2 Member Function Documentation

4.32.2.1 `void MTRandom::gen_state () [static, private]`

Definition at line 29 of file MTRandom.cpp.

References `p`, `state`, and `twiddle()`.

4.32.2.2 `double MTRandom::getDouble () [inline, static]`

Returns random double in $[0,1)$.

Definition at line 89 of file MTRandom.h.

4.32.2.3 `double MTRandom::getDoubleClosed () [inline, static]`

Returns random double in $[0,1]$.

Definition at line 94 of file MTRandom.h.

4.32.2.4 `double MTRandom::getDoubleHR () [inline, static]`

Returns 53 bit high resolution random double in $[0,1)$.

Definition at line 104 of file MTRandom.h.

4.32.2.5 `double MTRandom::getDoubleOpen () [inline, static]`

Returns random double in $(0,1)$.

Definition at line 99 of file MTRandom.h.

4.32.2.6 `float MTRandom::getFloat () [inline, static]`

Returns random float in $[0,1)$.

Definition at line 84 of file MTRandom.h.

4.32.2.7 `unsigned int MTRandom::getInt (unsigned int n) [inline, static]`

Returns unsigned int in $[0,n]$

Definition at line 79 of file MTRandom.h.

4.32.2.8 `unsigned long int MTRandom::getLongInt () [inline, static]`

Returns a random 32 bit int

Definition at line 67 of file MTRandom.h.

4.32.2.9 `void MTRandom::operator=(const MTRandom &) [private]`

4.32.2.10 `void MTRandom::seed (const unsigned long *, int size) [static]`

Definition at line 51 of file `MTRandom.cpp`.

References `p`, `seed()`, and `state`.

4.32.2.11 `void MTRandom::seed (unsigned long) [static]`

Definition at line 39 of file `MTRandom.cpp`.

References `p`, and `state`.

Referenced by `seed()`.

4.32.2.12 `unsigned long MTRandom::twiddle (unsigned long u, unsigned long v) [inline, static, private]`

Definition at line 117 of file `MTRandom.h`.

Referenced by `gen_state()`.

4.32.3 Member Data Documentation

4.32.3.1 `int MTRandom::p = 0 [static, private]`

Definition at line 16 of file `MTRandom.cpp`.

Referenced by `gen_state()`, and `seed()`.

4.32.3.2 `unsigned long MTRandom::state [static, private]`

Initial value:

```
{5489UL, 1301868182UL, 2938499221UL, 2950281878UL, 1875628136UL, 751856242UL, 944701696UL, 2243192071UL, 6
2277067795UL, 4131855432UL, 2722057515UL, 1264804546UL, 3848622725UL, 2211267957UL, 4100593547UL, 95912377
2802786495UL, 2728923319UL, 3996284304UL, 903417639UL, 1171249804UL, 1020374987UL, 2824535874UL, 423621996
3754009956UL, 3803931226UL, 4160647125UL, 1477814055UL, 4043852216UL, 1876372354UL, 3133294443UL, 38711048
2894533366UL, 3474657421UL, 2372634704UL, 2845748389UL, 43024175UL, 2774226648UL, 1987702864UL, 3186502468
}
```

Definition at line 10 of file `MTRandom.cpp`.

Referenced by `gen_state()`, and `seed()`.

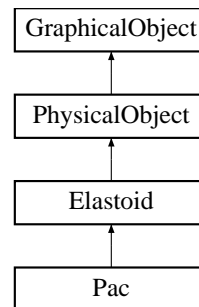
The documentation for this class was generated from the following files:

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

4.33 Pac Class Reference

```
#include <Pac.h>
```

Inheritance diagram for Pac::



Public Member Functions

- [Pac](#) ([real](#) x0, [real](#) y0)
- virtual [~Pac](#) ()
- virtual void [controll](#) ()

Public Attributes

- [ElasticLink](#) * [activeLink](#)
- [real](#) l
- [RandomController](#) controller

Private Member Functions

- int [index](#) (int i)

Private Attributes

- int [nCircles](#)
- [real](#) rPac
- [real](#) u0
- [real](#) u

4.33.1 Constructor & Destructor Documentation

4.33.1.1 Pac::Pac ([real](#) x0, [real](#) y0)

Definition at line 12 of file Pac.cpp.

References [activeLink](#), [Elastoid::addObject\(\)](#), [index\(\)](#), [ElasticLink::k](#), [l](#), [M_PI](#), [nCircles](#), [real](#), [rPac](#), [GraphicalObject::setColor\(\)](#), [u](#), and [u0](#).

4.33.1.2 **Pac::~~Pac ()** [virtual]

Definition at line 37 of file Pac.cpp.

4.33.2 Member Function Documentation

4.33.2.1 **void Pac::controll ()** [virtual]

Transmits perceptual information to the controller and gets motor information from it. To be overridden by agents or other objects with self-generated movement.

Reimplemented from [PhysicalObject](#).

Definition at line 41 of file Pac.cpp.

References [activeLink](#), [RandomController::advanceTime\(\)](#), [controller](#), [Controller::getOutput\(\)](#), [l](#), [ElasticLink::length](#), [M_PI](#), [nCircles](#), [Vector2::normalize\(\)](#), [real](#), [Vector2::rotate\(\)](#), [u](#), and [u0](#).

4.33.2.2 **int Pac::index (int i)** [inline, private]

Definition at line 36 of file Pac.h.

Referenced by [Pac\(\)](#).

4.33.3 Member Data Documentation

4.33.3.1 **ElasticLink* Pac::activeLink**

The link that can be controlled by the agent (the mouth).

Definition at line 17 of file Pac.h.

Referenced by [controll\(\)](#), and [Pac\(\)](#).

4.33.3.2 **RandomController Pac::controller**

Definition at line 23 of file Pac.h.

Referenced by [controll\(\)](#).

4.33.3.3 **real Pac::l**

The rest length of the active link.

Definition at line 20 of file Pac.h.

Referenced by [controll\(\)](#), and [Pac\(\)](#).

4.33.3.4 **int Pac::nCircles** [private]

The number of circles that compose the agent. Should be odd.

Definition at line 29 of file Pac.h.

Referenced by [controll\(\)](#), and [Pac\(\)](#).

4.33.3.5 [real Pac::rPac](#) [private]

The radius of the circular structure of the agent.

Definition at line 32 of file Pac.h.

Referenced by Pac().

4.33.3.6 [real Pac::u](#) [private]

Definition at line 34 of file Pac.h.

Referenced by controll(), and Pac().

4.33.3.7 [real Pac::u0](#) [private]

Definition at line 34 of file Pac.h.

Referenced by controll(), and Pac().

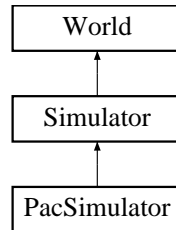
The documentation for this class was generated from the following files:

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

4.34 PacSimulator Class Reference

```
#include <PacSimulator.h>
```

Inheritance diagram for PacSimulator::



Public Member Functions

- [PacSimulator \(\)](#)
- virtual [~PacSimulator \(\)](#)

4.34.1 Constructor & Destructor Documentation

4.34.1.1 PacSimulator::PacSimulator ()

r, x, y.

Definition at line 14 of file PacSimulator.cpp.

References [degrees](#), [real](#), and [Simulator::registerObject\(\)](#).

4.34.1.2 PacSimulator::~~PacSimulator () [virtual]

Definition at line 46 of file PacSimulator.cpp.

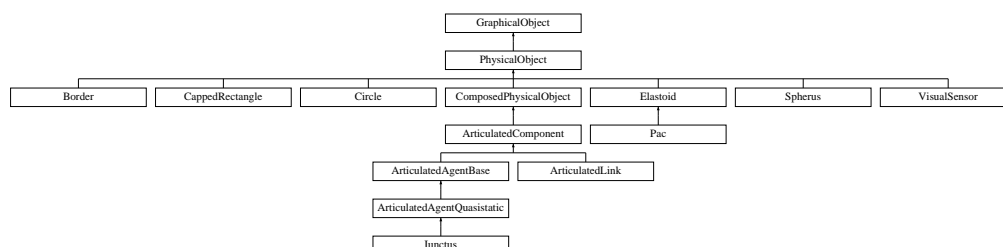
The documentation for this class was generated from the following files:

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

4.35 PhysicalObject Class Reference

```
#include <PhysicalObject.h>
```

Inheritance diagram for PhysicalObject::



Public Member Functions

- **PhysicalObject** (std::string label="", int nSensors=0, real saturationForce=0.5, Color outlineColor=GUI::colorBlack, Color fillColor=GUI::colorTransparent)
- virtual ~PhysicalObject ()
- virtual void registerPrimitives (Simulator *simulator)
- virtual bool detectContacts (PhysicalObject *object, GlobalContactInfoVector *contacts)=0
- virtual bool detectContacts (Border *object, GlobalContactInfoVector *contacts)
- virtual bool detectContacts (Circle *object, GlobalContactInfoVector *contacts)
- virtual bool detectContacts (CappedRectangle *object, GlobalContactInfoVector *contacts)
- void setPosition (real x, real y)
- void setVelocity (real vx, real vy)
- void setMass (real m)
- virtual void computeMass (real density=ThyrixParameters::defaultDensity)
- virtual void computeInertia ()
- virtual int isCircle ()
- void normalizeAlpha ()
- virtual void integrate (const Integrator &integrator)
- virtual void rollback ()
- virtual void computeBox ()
- virtual void computeDerivativesWithoutContacts (ContactSolver *contactSolver)
- virtual void computeDerivatives (GlobalContactInfoVector *globalContacts)
- virtual void setSensors ()
- virtual void setSensor (ContactInfo *contact)
- virtual void controll ()
- virtual void detectInternalContacts (GlobalContactInfoVector *globalContacts)
- virtual void deleteContacts ()
- virtual void resetSensors ()
- void addContact (ContactInfo *contact)
- virtual void fillContactMatrix (ContactSolver *contactSolver, ContactInfo *contact)
- void drawContactForces (GUI *gui)
- virtual bool detectMouseContact (const Vector2 &rMouse, Vector2 &p, PhysicalObject *&object)

Public Attributes

- [Vector2](#) r
- [Vector2](#) rOld
- [Vector2](#) v
- [real](#) m
- [real](#) alpha
- [real](#) alphaOld
- [real](#) omega
- [real](#) I
- [Vector2](#) externalForce
- [real](#) externalTorque
- [PhysicalObject](#) * parent
- [Vector2](#) relativeR
- [real](#) relativeAlpha
- [Vector2](#) boxMin
- [Vector2](#) boxMax
- [ContactInfoPVector](#) contacts
- [std::string](#) label
- [real](#) * activations
- [real](#) saturationForce

Protected Attributes

- [int](#) nSensors

4.35.1 Detailed Description

A physical object has the following properties: position r, velocity v, mass m, angle alpha, angular velocity omega. The linear properties are the properties of the center of mass, relative to the laboratory reference system. The rotational properties refer to the rotation around the z axis that passes through the center of mass of the object, relative to the laboratory reference system.

The object can have tactile sensors distributed on its surface. nSensors is the number of tactile sensors. The tactile sensors have a maximum activation (1) for a force greater than saturationForce.

Definition at line 26 of file PhysicalObject.h.

4.35.2 Constructor & Destructor Documentation

4.35.2.1 `PhysicalObject::PhysicalObject (std::string label = "", int nSensors = 0, real saturationForce = 0.5, Color outlineColor = GUI::colorBlack, Color fillColor = GUI::colorTransparent)`

Default constructor; initializes everything with 0.

Definition at line 8 of file PhysicalObject.cpp.

References activations, alpha, boxMax, boxMin, externalTorque, I, m, omega, parent, real, relativeAlpha, and [Vector2::setXY\(\)](#).

4.35.2.2 PhysicalObject::~~PhysicalObject () [virtual]

Destructor; purges the lists.

Definition at line 33 of file PhysicalObject.cpp.

References activations, and deleteContacts().

4.35.3 Member Function Documentation**4.35.3.1 void PhysicalObject::addContact (ContactInfo * contact)** [inline]

Adds given contact to the contact list.

Definition at line 116 of file PhysicalObject.h.

Referenced by ContactInfo::setupObject().

4.35.3.2 virtual void PhysicalObject::computeBox () [inline, virtual]

Computes the coordinates of the bounding box. It is called after each call to [integrate\(\)](#) or [rollback\(\)](#).

Reimplemented in [CappedRectangle](#), and [Circle](#).

Definition at line 83 of file PhysicalObject.h.

Referenced by ComposedPhysicalObject::computeMemberPositions(), [integrate\(\)](#), and [rollback\(\)](#).

4.35.3.3 void PhysicalObject::computeDerivatives (GlobalContactInfoVector * globalContacts)
[virtual]

Computes the derivatives, given the contact forces. The globalContacts pointer is needed for contact processing for articulated agents.

Reimplemented in [Spherus](#), [Elastoid](#), and [ArticulatedAgentQuasistatic](#).

Definition at line 80 of file PhysicalObject.cpp.

References contacts, ContactInfo::force, ContactInfo::n, omega, ContactInfo::pxn, setSensors(), ContactInfo::sigma, and v.

Referenced by [Spherus::computeDerivatives\(\)](#).

4.35.3.4 void PhysicalObject::computeDerivativesWithoutContacts (ContactSolver * contactSolver) [virtual]

Computes the derivatives (velocity, angular velocity) of the object, ignoring any eventual contact forces.

Reimplemented in [Spherus](#), [Border](#), [Elastoid](#), and [ArticulatedAgentQuasistatic](#).

Definition at line 71 of file PhysicalObject.cpp.

References externalForce, externalTorque, omega, Vector2::setToZero(), and v.

Referenced by [Spherus::computeDerivativesWithoutContacts\(\)](#).

4.35.3.5 **virtual void PhysicalObject::computeInertia ()** [inline, virtual]

Computes the moment of inertia around the z axis that passes through the center, based on the mass and the geometrical properties. To be overridden.

Reimplemented in [CappedRectangle](#), and [Circle](#).

Definition at line 65 of file PhysicalObject.h.

References I.

4.35.3.6 **virtual void PhysicalObject::computeMass (real density = ThyrixParameters::defaultDensity)** [inline, virtual]

Sets the mass based on the density. To be overridden.

Reimplemented in [CappedRectangle](#), and [Circle](#).

Definition at line 61 of file PhysicalObject.h.

References m.

4.35.3.7 **virtual void PhysicalObject::controll ()** [inline, virtual]

Transmits perceptual information to the controller and gets motor information from it. To be overridden by agents or other objects with self-generated movement.

Reimplemented in [Iunctus](#), [Spherus](#), and [Pac](#).

Definition at line 102 of file PhysicalObject.h.

Referenced by Simulator::controll().

4.35.3.8 **void PhysicalObject::deleteContacts ()** [virtual]

Delete the contacts of the object.

Reimplemented in [Iunctus](#), [Spherus](#), [Elastoid](#), [ArticulatedAgentBase](#), and [ArticulatedLink](#).

Definition at line 43 of file PhysicalObject.cpp.

References contacts, purgeContainer(), and resetSensors().

Referenced by [Spherus::deleteContacts\(\)](#), [Iunctus::deleteContacts\(\)](#), [ArticulatedLink::deleteContacts\(\)](#), [ArticulatedAgentBase::deleteContacts\(\)](#), and [~PhysicalObject\(\)](#).

4.35.3.9 **virtual bool PhysicalObject::detectContacts (CappedRectangle * object, GlobalContactInfoVector * contacts)** [inline, virtual]

Reimplemented in [CappedRectangle](#), and [VisualSensor](#).

Definition at line 47 of file PhysicalObject.h.

References detectContacts().

4.35.3.10 `virtual bool PhysicalObject::detectContacts (Circle * object, GlobalContactInfoVector * contacts)` [inline, virtual]

Reimplemented in [CappedRectangle](#), [Circle](#), and [VisualSensor](#).

Definition at line 44 of file [PhysicalObject.h](#).

References [detectContacts\(\)](#).

4.35.3.11 `virtual bool PhysicalObject::detectContacts (Border * object, GlobalContactInfoVector * contacts)` [inline, virtual]

Reimplemented in [Border](#), [CappedRectangle](#), [Circle](#), and [VisualSensor](#).

Definition at line 41 of file [PhysicalObject.h](#).

References [detectContacts\(\)](#).

4.35.3.12 `virtual bool PhysicalObject::detectContacts (PhysicalObject * object, GlobalContactInfoVector * contacts)` [pure virtual]

Contact handling.

Implemented in [Spherus](#), [Border](#), [CappedRectangle](#), [Circle](#), [ComposedPhysicalObject](#), [Elastoid](#), [VisualSensor](#), and [ArticulatedAgentBase](#).

Referenced by [VisualSensor::detectContacts\(\)](#), [Simulator::detectContacts\(\)](#), [detectContacts\(\)](#), [Circle::detectContacts\(\)](#), [CappedRectangle::detectContacts\(\)](#), and [Border::detectContacts\(\)](#).

4.35.3.13 `virtual void PhysicalObject::detectInternalContacts (GlobalContactInfoVector * globalContacts)` [inline, virtual]

Detects internal contacts. Does nothing for primitive objects or composed objects. To be overridden by complex objects that have parts that move relative to each other, such as articulated objects.

Reimplemented in [Spherus](#), [Elastoid](#), and [ArticulatedAgentBase](#).

Definition at line 107 of file [PhysicalObject.h](#).

4.35.3.14 `virtual bool PhysicalObject::detectMouseContact (const Vector2 & rMouse, Vector2 & p, PhysicalObject *& object)` [inline, virtual]

Detects whether a click of the mouse at *rMouse* has touched the object. In case of contact, returns true and sets *p* to the relative vector between the center of the object and the contact point; *p* is expressed in the object reference frame because it rotates with the object while the object is dragged. The object needs to be set by the function because, for articulated objects, the dragged object is an articulation and not the whole articulated object. To be overridden by the various object primitives.

Reimplemented in [Spherus](#), [CappedRectangle](#), [Circle](#), [ComposedPhysicalObject](#), [Elastoid](#), and [ArticulatedAgentBase](#).

Definition at line 133 of file [PhysicalObject.h](#).

4.35.3.15 void PhysicalObject::drawContactForces (GUI * gui)

Draws the contact forces.

Definition at line 135 of file PhysicalObject.cpp.

References `contacts`, `GUI::drawForce()`, `ContactInfo::force`, `ContactInfo::n`, `ContactInfo::p`, `r`, and `ContactInfo::sigma`.

4.35.3.16 void PhysicalObject::fillContactMatrix (ContactSolver * contactSolver, ContactInfo * contact) [virtual]

Computes the contribution of the current object to the matrix of interactions between all objects that are in contact.

Reimplemented in `Border`, `ArticulatedAgentBase`, and `ArticulatedLink`.

Definition at line 95 of file PhysicalObject.cpp.

References `ContactInfo::alpha`, `ContactSolver::contactMatrix`, `contacts`, `ContactSolver::contactVector`, `Vector2::cross()`, `ContactInfo::n`, `omega`, `ContactInfo::p`, `ContactInfo::pxn`, and `ContactInfo::sigma`.

4.35.3.17 void PhysicalObject::integrate (const Integrator & integrator) [virtual]

Advances the time to the next timestep.

Reimplemented in `Spherus`, `Border`, `ComposedPhysicalObject`, `Elastoid`, `ArticulatedAgentQuasistatic`, and `ArticulatedLink`.

Definition at line 56 of file PhysicalObject.cpp.

References `alpha`, `alphaOld`, `computeBox()`, `Integrator::integrate()`, `normalizeAlpha()`, `omega`, `r`, `rOld`, and `v`.

Referenced by `Spherus::integrate()`, and `ComposedPhysicalObject::integrate()`.

4.35.3.18 virtual int PhysicalObject::isCircle () [inline, virtual]

Verifies if the object has circular symmetry; in this case, computations of rotational properties can be skipped, on various occasions. By default, returns 0 (not symmetrical). To be overridden by symmetrical objects (`Circle`).

Reimplemented in `Circle`.

Definition at line 70 of file PhysicalObject.h.

4.35.3.19 void PhysicalObject::normalizeAlpha () [inline]

Normalizes Alpha to a value between 0 and 2 PI.

Definition at line 203 of file PhysicalObject.h.

References `alpha`.

Referenced by `integrate()`.

4.35.3.20 void PhysicalObject::registerPrimitives ([Simulator](#) * *simulator*) [virtual]

Registers all composing primitives of the object with the simulator.

Reimplemented in [ComposedPhysicalObject](#), [Elastoid](#), and [ArticulatedAgentBase](#).

Definition at line 39 of file [PhysicalObject.cpp](#).

References [Simulator::registerPrimitive\(\)](#).

Referenced by [Simulator::registerObject\(\)](#).

4.35.3.21 void PhysicalObject::resetSensors () [virtual]

Resets tactile sensors to zero.

Reimplemented in [ComposedPhysicalObject](#).

Definition at line 48 of file [PhysicalObject.cpp](#).

References [activations](#).

Referenced by [deleteContacts\(\)](#), and [ComposedPhysicalObject::resetSensors\(\)](#).

4.35.3.22 void PhysicalObject::rollback () [virtual]

Rolls back the time to the previous timestep.

Reimplemented in [Border](#), [ComposedPhysicalObject](#), [ArticulatedAgentQuasistatic](#), and [ArticulatedLink](#).

Definition at line 65 of file [PhysicalObject.cpp](#).

References [alpha](#), [computeBox\(\)](#), and [r](#).

Referenced by [ComposedPhysicalObject::rollback\(\)](#).

4.35.3.23 void PhysicalObject::setMass ([real](#) *m*) [inline]

Sets the mass *m* of the object.

Definition at line 58 of file [PhysicalObject.h](#).

4.35.3.24 void PhysicalObject::setPosition ([real](#) *x*, [real](#) *y*) [inline]

Sets the position *r* of the object.

Definition at line 52 of file [PhysicalObject.h](#).

References [r](#), [Vector2::x](#), and [Vector2::y](#).

4.35.3.25 virtual void PhysicalObject::setSensor ([ContactInfo](#) * *contact*) [inline, virtual]

Sets the activation of a tactile sensor, corresponding to the given contact. To be overridden.

Reimplemented in [CappedRectangle](#), and [Circle](#).

Definition at line 98 of file [PhysicalObject.h](#).

Referenced by [setSensors\(\)](#), and [ComposedPhysicalObject::setSensors\(\)](#).

4.35.3.26 void PhysicalObject::setSensors () [virtual]

Sets the activations of the tactile sensors, on the basis of the contact forces related to the contacts of the object with other objects.

Reimplemented in [ComposedPhysicalObject](#).

Definition at line 126 of file [PhysicalObject.cpp](#).

References [contacts](#), and [setSensor\(\)](#).

Referenced by [computeDerivatives\(\)](#).

4.35.3.27 void PhysicalObject::setVelocity (real vx, real vy) [inline]

Sets the velocity v of the object.

Definition at line 55 of file [PhysicalObject.h](#).

References [v](#), [Vector2::x](#), and [Vector2::y](#).

4.35.4 Member Data Documentation**4.35.4.1 real* PhysicalObject::activations**

A vector containing the activations of the sensors.

Definition at line 193 of file [PhysicalObject.h](#).

Referenced by [PhysicalObject\(\)](#), [resetSensors\(\)](#), and [~PhysicalObject\(\)](#).

4.35.4.2 real PhysicalObject::alpha

Rotation angle around center of mass relative to the LRS

Definition at line 149 of file [PhysicalObject.h](#).

Referenced by [Simulator::applyMouseForce\(\)](#), [ArticulatedAgentQuasistatic::backwardDynamics\(\)](#), [VisualSensor::computeGamma\(\)](#), [ComposedPhysicalObject::computeMemberPositions\(\)](#), [VisualSensor::draw\(\)](#), [Iunctus::draw\(\)](#), [ArticulatedAgentQuasistatic::forwardAccelerations\(\)](#), [ArticulatedAgentQuasistatic::forwardKinematics\(\)](#), [Simulator::indexContacts\(\)](#), [Spherus::integrate\(\)](#), [integrate\(\)](#), [normalizeAlpha\(\)](#), [PhysicalObject\(\)](#), and [rollback\(\)](#).

4.35.4.3 real PhysicalObject::alphaOld

Value of alpha at the previous timestep.

Definition at line 152 of file [PhysicalObject.h](#).

Referenced by [ComposedPhysicalObject::computeMemberPositions\(\)](#), and [integrate\(\)](#).

4.35.4.4 Vector2 PhysicalObject::boxMax

Minimum coordinates of the axis-aligned bounding box surrounding the object.

Definition at line 184 of file [PhysicalObject.h](#).

Referenced by `ComposedPhysicalObject::computeMemberPositions()`, `ArticulatedAgentQuasistatic::integrate()`, `PhysicalObject()`, and `ArticulatedAgentQuasistatic::rollback()`.

4.35.4.5 **Vector2** `PhysicalObject::boxMin`

Minimum coordinates of the axis-aligned bounding box surrounding the object.

Definition at line 181 of file `PhysicalObject.h`.

Referenced by `ComposedPhysicalObject::computeMemberPositions()`, `ArticulatedAgentQuasistatic::integrate()`, `PhysicalObject()`, `ArticulatedAgentQuasistatic::rollback()`, and `Simulator::sortObjects()`.

4.35.4.6 **ContactInfoPVector** `PhysicalObject::contacts`

The list of contacts that act on the object. The list owns the contacts.

Definition at line 187 of file `PhysicalObject.h`.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, `computeDerivatives()`, `deleteContacts()`, `drawContactForces()`, `fillContactMatrix()`, `ArticulatedAgentQuasistatic::forwardAccelerations()`, and `setSensors()`.

4.35.4.7 **Vector2** `PhysicalObject::externalForce`

The sum of external forces acting on the object

Definition at line 161 of file `PhysicalObject.h`.

Referenced by `ElasticLink::applyForces()`, `Simulator::applyMouseForce()`, `ArticulatedAgentQuasistatic::backwardDynamics()`, `Spherus::computeDerivativesWithoutContacts()`, and `computeDerivativesWithoutContacts()`.

4.35.4.8 **real** `PhysicalObject::externalTorque`

The sum of external torques acting on the object

Definition at line 164 of file `PhysicalObject.h`.

Referenced by `Simulator::applyMouseForce()`, `ArticulatedAgentQuasistatic::backwardDynamics()`, `computeDerivativesWithoutContacts()`, and `PhysicalObject()`.

4.35.4.9 **real** `PhysicalObject::I`

Moment of inertia for rotation around the z axis that passes through the center of mass.

Definition at line 158 of file `PhysicalObject.h`.

Referenced by `computeInertia()`, `ComposedPhysicalObject::computeMassProperties()`, and `PhysicalObject()`.

4.35.4.10 **std::string** `PhysicalObject::label`

A text label for the object; used primarily for debugging.

Definition at line 190 of file `PhysicalObject.h`.

Referenced by `ComposedPhysicalObject::addObject()`.

4.35.4.11 `real PhysicalObject::m`

Mass

Definition at line 146 of file `PhysicalObject.h`.

Referenced by `Spherus::computeCircleProprioception()`, `computeMass()`, `ComposedPhysicalObject::computeMassProperties()`, and `PhysicalObject()`.

4.35.4.12 `int PhysicalObject::nSensors` [protected]

The number of unit sensors (pixels) in the tactile sensor.

Reimplemented in `Spherus`.

Definition at line 200 of file `PhysicalObject.h`.

4.35.4.13 `real PhysicalObject::omega`

Angular velocity around center of mass relative to the LRS

Definition at line 155 of file `PhysicalObject.h`.

Referenced by `computeDerivatives()`, `computeDerivativesWithoutContacts()`, `fillContactMatrix()`, `ArticulatedAgentQuasistatic::forwardAccelerations()`, `integrate()`, and `PhysicalObject()`.

4.35.4.14 `PhysicalObject* PhysicalObject::parent`

The parent of the object. It is NULL for simple objects, not NULL for objects that are members of composed objects.

Definition at line 168 of file `PhysicalObject.h`.

Referenced by `ComposedPhysicalObject::addObject()`, `PhysicalObject()`, and `ContactInfo::setupObject()`.

4.35.4.15 `Vector2 PhysicalObject::r`

Position of the center of mass of the object relative to laboratory reference system (LRS)

Definition at line 137 of file `PhysicalObject.h`.

Referenced by `ElasticLink::applyForces()`, `Simulator::applyMouseForce()`, `Spherus::computeMemberPositions()`, `ComposedPhysicalObject::computeMemberPositions()`, `Spherus::computeState()`, `VisualSensor::detectContacts()`, `Circle::detectContacts()`, `CappedRectangle::detectContacts()`, `VisualSensor::draw()`, `Spherus::draw()`, `Iunctus::draw()`, `ElasticLink::draw()`, `drawContactForces()`, `ArticulatedAgentQuasistatic::forwardKinematics()`, `integrate()`, `rollback()`, `setPosition()`, and `ContactInfo::setupObject()`.

4.35.4.16 [real PhysicalObject::relativeAlpha](#)

The rotation of the member object relative to the reference system of the composed object (ORS), for objects that are members of composed objects.

Definition at line 178 of file PhysicalObject.h.

Referenced by `ComposedPhysicalObject::addObject()`, `ComposedPhysicalObject::computeMemberPositions()`, and `PhysicalObject()`.

4.35.4.17 [Vector2 PhysicalObject::relativeR](#)

The position of the member object relative to the reference system of the composed object (ORS), for objects that are members of composed objects.

Definition at line 173 of file PhysicalObject.h.

Referenced by `ComposedPhysicalObject::addObject()`, `ComposedPhysicalObject::computeMassProperties()`, and `ComposedPhysicalObject::computeMemberPositions()`.

4.35.4.18 [Vector2 PhysicalObject::rOld](#)

Value of r at the previous timestep.

Definition at line 140 of file PhysicalObject.h.

Referenced by `ComposedPhysicalObject::computeMemberPositions()`, and `integrate()`.

4.35.4.19 [real PhysicalObject::saturationForce](#)

The saturation contact force for contact sensors

Definition at line 196 of file PhysicalObject.h.

4.35.4.20 [Vector2 PhysicalObject::v](#)

Velocity relative to the LRS

Definition at line 143 of file PhysicalObject.h.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, `Spherus::computeCircleProprioception()`, `computeDerivatives()`, `computeDerivativesWithoutContacts()`, `ArticulatedAgentQuasistatic::forwardAccelerations()`, `integrate()`, and `setVelocity()`.

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

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

4.36 Random Class Reference

```
#include <Random.h>
```

Public Member Functions

- [Random](#) ()
- virtual [~Random](#) ()

Static Public Member Functions

- void [setSeed](#) (unsigned int seed=0)
- float [getFloat](#) ()
- double [getDouble](#) ()
- unsigned int [getInt](#) (unsigned int n)

4.36.1 Constructor & Destructor Documentation

4.36.1.1 Random::Random ()

Definition at line 4 of file Random.cpp.

4.36.1.2 Random::~~Random () [virtual]

Definition at line 8 of file Random.cpp.

4.36.2 Member Function Documentation

4.36.2.1 double Random::getDouble () [inline, static]

Returns random double in [0,1).

Definition at line 20 of file Random.h.

4.36.2.2 float Random::getFloat () [inline, static]

Returns random float in [0,1).

Definition at line 14 of file Random.h.

Referenced by RandomController::advanceTime(), and RandomController::RandomController().

4.36.2.3 unsigned int Random::getInt (unsigned int *n*) [inline, static]

Returns unsigned int in [0,n]

Definition at line 25 of file Random.h.

4.36.2.4 void Random::setSeed (unsigned int *seed* = 0) [static]

Sets the random number generator seed. If seed is 0, sets seed based on current time.

Definition at line 12 of file Random.cpp.

Referenced by RandomController::RandomController().

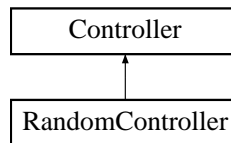
The documentation for this class was generated from the following files:

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

4.37 RandomController Class Reference

```
#include <RandomController.h>
```

Inheritance diagram for RandomController::



Public Member Functions

- [RandomController](#) (unsigned int [nInputs](#), unsigned int [nOutputs](#), unsigned int randSeed=0)
- virtual [~RandomController](#) ()
- virtual void [advanceTime](#) ()

4.37.1 Constructor & Destructor Documentation

4.37.1.1 RandomController::RandomController (unsigned int *nInputs*, unsigned int *nOutputs*, unsigned int *randSeed* = 0)

Definition at line 4 of file RandomController.cpp.

References [Random::getFloat\(\)](#), and [Random::setSeed\(\)](#).

4.37.1.2 RandomController::~~RandomController () [virtual]

Definition at line 12 of file RandomController.cpp.

4.37.2 Member Function Documentation

4.37.2.1 void RandomController::advanceTime () [virtual]

Reimplemented from [Controller](#).

Definition at line 16 of file RandomController.cpp.

References [Random::getFloat\(\)](#).

Referenced by [Pac::controll\(\)](#), and [Iunctus::controll\(\)](#).

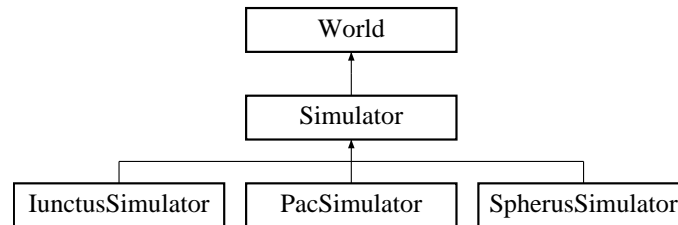
The documentation for this class was generated from the following files:

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

4.38 Simulator Class Reference

```
#include <Simulator.h>
```

Inheritance diagram for Simulator::



Public Member Functions

- [Simulator](#) ()
- virtual [~Simulator](#) ()
- virtual double [getDt](#) () const
- virtual void [advanceTime](#) ()
- void [registerObject](#) ([PhysicalObject](#) *object)
- void [registerPrimitive](#) ([PhysicalObject](#) *object)
- void [setup](#) ()
- void [indexContacts](#) ()
- void [fillContactMatrix](#) ()
- virtual void [detectContacts](#) ()
- void [computeContacts](#) ()
- virtual void [draw](#) ([GUI](#) *gui)
- void [setMouseForce](#) (float x, float y)
- void [unsetMouseForce](#) ()
- virtual void [onMouseLeftDown](#) (float x, float y)
- virtual void [onMouseLeftUp](#) (float x, float y)
- void [applyMouseForce](#) ()

Public Attributes

- [PhysicalObject](#) * [draggedObject](#)
- [Vector2](#) [draggingPoint](#)
- [ContactSolver](#) * [contactSolver](#)
- [PhysicalObjectPVector](#) [objects](#)
- [PhysicalObjectPVector](#) [primitives](#)
- [GlobalContactInfoVector](#) [contacts](#)

Protected Member Functions

- virtual void [controll](#) ()

Protected Attributes

- [Integrator](#) `integrator`
- `real` `realTime`

Private Member Functions

- `void` [deleteObjects](#) ()
- `void` [deleteContacts](#) ()
- `void` [sortObjects](#) (int n)

Private Attributes

- `bool` [isContact](#)
- [PhysicalObjectPVector](#) `sortedObjects` [2]
- `std::set< ObjectPair >` `xContacts`
- `std::set< ObjectPair >` `xyContacts`

4.38.1 Detailed Description

The main class of the Thyrix simulator.

Definition at line 13 of file `Simulator.h`.

4.38.2 Constructor & Destructor Documentation

4.38.2.1 `Simulator::Simulator ()`

Definition at line 9 of file `Simulator.cpp`.

References `contacts`, `contactSolver`, `dtDefault`, `isContact`, and `realTime`.

4.38.2.2 `Simulator::~~Simulator ()` [virtual]

Definition at line 23 of file `Simulator.cpp`.

References `deleteObjects()`.

4.38.3 Member Function Documentation

4.38.3.1 `void Simulator::advanceTime ()` [virtual]

Performs a timestep. It needs to be virtual because of the use of abstract `Simulator` classes in GUI-related base classes. This method may be overridden to include fitness computing, agent tracking etc.

Implements [World](#).

Definition at line 51 of file `Simulator.cpp`.

References `applyMouseForce()`, `ContactSolver::computeContacts()`, `contacts`, `contactSolver`, `controll()`, `deleteContacts()`, `detectContacts()`, `fillContactMatrix()`, `Integrator::getDt()`, `indexContacts()`, `integrator`, `isContact`, `objects`, `realTime`, and `ContactSolver::uploadForces()`.

4.38.3.2 void Simulator::applyMouseForce ()

Adds the mouse force to the external forces (and torques) of the object dragged with the mouse.

Definition at line 262 of file Simulator.cpp.

References `PhysicalObject::alpha`, `draggedObject`, `PhysicalObject::externalForce`, `PhysicalObject::externalTorque`, `PhysicalObject::r`, `Vector2::rotate()`, `Vector2::x`, and `Vector2::y`.

Referenced by `advanceTime()`.

4.38.3.3 void Simulator::computeContacts ()

Definition at line 225 of file Simulator.cpp.

References `ContactSolver::computeContacts()`, and `contactSolver`.

4.38.3.4 void Simulator::controll () [protected, virtual]

Definition at line 45 of file Simulator.cpp.

References `PhysicalObject::controll()`, and `objects`.

Referenced by `advanceTime()`.

4.38.3.5 void Simulator::deleteContacts () [private]

Deletes all previous contact information: the global vector of contacts, and the contacts of all objects.

Definition at line 229 of file Simulator.cpp.

References `contacts`, and `objects`.

Referenced by `advanceTime()`.

4.38.3.6 void Simulator::deleteObjects () [private]

Deletes the object list and the included objects.

Definition at line 41 of file Simulator.cpp.

References `objects`, and `purgeContainer()`.

Referenced by `~Simulator()`.

4.38.3.7 void Simulator::detectContacts () [virtual]

Reimplemented in [IunctusSimulator](#).

Definition at line 135 of file Simulator.cpp.

References `contacts`, `PhysicalObject::detectContacts()`, `Simulator::ObjectPair::o1`, `sortedObjects`, `sortObjects()`, `xContacts`, and `xyContacts`.

Referenced by `advanceTime()`, `IunctusSimulator::detectContacts()`, and `ComposedPhysicalObject::detectContacts()`.

4.38.3.8 void Simulator::draw (GUI * gui) [virtual]

Reimplemented from [World](#).

Definition at line 279 of file Simulator.cpp.

References [World::draw\(\)](#), [objects](#), and [GUI::setBrushColor\(\)](#).

4.38.3.9 void Simulator::fillContactMatrix ()

Computes the matrix used for contact force computation.

Definition at line 212 of file Simulator.cpp.

References [contacts](#), and [contactSolver](#).

Referenced by [advanceTime\(\)](#).

4.38.3.10 virtual double Simulator::getDt () const [inline, virtual]

Reimplemented from [World](#).

Definition at line 18 of file Simulator.h.

References [Integrator::getDt\(\)](#), and [integrator](#).

4.38.3.11 void Simulator::indexContacts ()

Indexes the contacts, resets to zero the contact data structures, allocates memory for them if needed.

Definition at line 198 of file Simulator.cpp.

References [PhysicalObject::alpha](#), [contacts](#), [contactSolver](#), and [ContactSolver::init\(\)](#).

Referenced by [advanceTime\(\)](#).

4.38.3.12 virtual void Simulator::onMouseLeftDown (float x, float y) [inline, virtual]

Reimplemented from [World](#).

Definition at line 61 of file Simulator.h.

References [World::onMouseLeftDown\(\)](#), and [setMouseForce\(\)](#).

4.38.3.13 virtual void Simulator::onMouseLeftUp (float x, float y) [inline, virtual]

Reimplemented from [World](#).

Definition at line 66 of file Simulator.h.

References [World::onMouseLeftUp\(\)](#), and [unsetMouseForce\(\)](#).

4.38.3.14 void Simulator::registerObject (PhysicalObject * object)

Adds the object to the objects array.

Definition at line 30 of file Simulator.cpp.

References objects, `PhysicalObject::registerPrimitives()`, and `sortedObjects`.

Referenced by `IunctusSimulator::IunctusSimulator()`, `PacSimulator::PacSimulator()`, and `SpherusSimulator::SpherusSimulator()`.

4.38.3.15 `void Simulator::registerPrimitive (PhysicalObject * object)`

Adds the object to the primitives array.

Definition at line 37 of file `Simulator.cpp`.

References primitives.

Referenced by `PhysicalObject::registerPrimitives()`, `Elastoid::registerPrimitives()`, and `ComposedPhysicalObject::registerPrimitives()`.

4.38.3.16 `void Simulator::setMouseForce (float x, float y)`

Definition at line 236 of file `Simulator.cpp`.

References `draggedObject`, `draggingPoint`, and `objects`.

Referenced by `onMouseLeftDown()`.

4.38.3.17 `void Simulator::setup ()` [inline]

Needs to be called after all objects were registered by the simulator, if we use box sorting for broad phase contact detection.

Definition at line 33 of file `Simulator.h`.

4.38.3.18 `void Simulator::sortObjects (int n)` [private]

Sorts the $n=0,1$ list of sorted objects, with insertion sort.

Definition at line 173 of file `Simulator.cpp`.

References `PhysicalObject::boxMin`, and `sortedObjects`.

Referenced by `detectContacts()`.

4.38.3.19 `void Simulator::unsetMouseForce ()`

Definition at line 252 of file `Simulator.cpp`.

References `draggedObject`.

Referenced by `onMouseLeftUp()`.

4.38.4 Member Data Documentation

4.38.4.1 `GlobalContactInfoVector Simulator::contacts`

The global list of contacts. This is a list of objects, not a list of pointers.

Definition at line 88 of file `Simulator.h`.

Referenced by `advanceTime()`, `deleteContacts()`, `detectContacts()`, `fillContactMatrix()`, `indexContacts()`, and `Simulator()`.

4.38.4.2 **ContactSolver*** `Simulator::contactSolver`

Definition at line 75 of file `Simulator.h`.

Referenced by `advanceTime()`, `computeContacts()`, `fillContactMatrix()`, `indexContacts()`, and `Simulator()`.

4.38.4.3 **PhysicalObject*** `Simulator::draggedObject`

The object dragged by the mouse.

Definition at line 53 of file `Simulator.h`.

Referenced by `applyMouseForce()`, `setMouseForce()`, and `unsetMouseForce()`.

4.38.4.4 **Vector2** `Simulator::draggingPoint`

Definition at line 54 of file `Simulator.h`.

Referenced by `setMouseForce()`.

4.38.4.5 **Integrator** `Simulator::integrator` [protected]

Definition at line 92 of file `Simulator.h`.

Referenced by `advanceTime()`, and `getDt()`.

4.38.4.6 **bool** `Simulator::isContact` [private]

Definition at line 106 of file `Simulator.h`.

Referenced by `advanceTime()`, and `Simulator()`.

4.38.4.7 **PhysicalObjectPVector** `Simulator::objects`

The list of top level objects in the environment (borders, ball, agents, etc.). This list owns the objects. The top level objects may contain member objects, that they own.

Definition at line 80 of file `Simulator.h`.

Referenced by `advanceTime()`, `controll()`, `deleteContacts()`, `deleteObjects()`, `draw()`, `registerObject()`, and `setMouseForce()`.

4.38.4.8 **PhysicalObjectPVector** `Simulator::primitives`

The list of low level objects (primitives) in the environment. The list does not own the objects. Currently, not used.

Definition at line 84 of file `Simulator.h`.

Referenced by `registerPrimitive()`.

4.38.4.9 [real Simulator::realTime](#) [protected]

Definition at line 96 of file Simulator.h.

Referenced by `advanceTime()`, and `Simulator()`.

4.38.4.10 [PhysicalObjectPVector Simulator::sortedObjects\[2\]](#) [private]

Lists of objects used for contact detection. The 2 lists correspond to the x and respectively y coordinates used for sorting.

Definition at line 110 of file Simulator.h.

Referenced by `detectContacts()`, `registerObject()`, and `sortObjects()`.

4.38.4.11 [std::set<ObjectPair> Simulator::xContacts](#) [private]

Definition at line 131 of file Simulator.h.

Referenced by `detectContacts()`.

4.38.4.12 [std::set<ObjectPair> Simulator::xyContacts](#) [private]

Definition at line 131 of file Simulator.h.

Referenced by `detectContacts()`.

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

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

4.39 Simulator::ObjectPair Class Reference

Public Member Functions

- [ObjectPair](#) ()
- [ObjectPair](#) ([PhysicalObject](#) *no1, [PhysicalObject](#) *no2)
- bool [operator<](#) (const [ObjectPair](#) &p2) const

Public Attributes

- [PhysicalObject](#) * o1
- [PhysicalObject](#) * o2

4.39.1 Detailed Description

A pair of objects: used for contact detection.

Definition at line 115 of file Simulator.h.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 Simulator::ObjectPair::ObjectPair () [inline]

Definition at line 117 of file Simulator.h.

4.39.2.2 Simulator::ObjectPair::ObjectPair ([PhysicalObject](#) * no1, [PhysicalObject](#) * no2) [inline]

Definition at line 118 of file Simulator.h.

4.39.3 Member Function Documentation

4.39.3.1 bool Simulator::ObjectPair::operator< (const [ObjectPair](#) & p2) const [inline]

Definition at line 124 of file Simulator.h.

References o1, and o2.

4.39.4 Member Data Documentation

4.39.4.1 [PhysicalObject*](#) Simulator::ObjectPair::o1

Definition at line 122 of file Simulator.h.

Referenced by Simulator::detectContacts(), and operator<().

4.39.4.2 [PhysicalObject*](#) [Simulator::ObjectPair::o2](#)

Definition at line 123 of file Simulator.h.

Referenced by [operator<\(\)](#).

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

- [Simulator.h](#)

4.40 SimulatorThread Class Reference

```
#include <SimulatorThread.h>
```

Public Member Functions

- [SimulatorThread](#) ([ThyrixMainFrame](#) *iniFrame)
- virtual [~SimulatorThread](#) ()
- virtual void * [Entry](#) ()
- virtual void [OnExit](#) ()
- void [setTimeFactor](#) (float newFactor)
- void [setFramesPerSecond](#) (int fps)
- void [setPause](#) ()
- void [step](#) ()

Public Attributes

- [World](#) * world

Private Member Functions

- void [sleepIfAhead](#) ()

Private Attributes

- float [timeFactor](#)
- unsigned [frameInterval](#)
- bool [paused](#)
- float [expectedTime](#)
- wxStopWatch [simulationTimer](#)
- [ThyrixMainFrame](#) * frame

4.40.1 Constructor & Destructor Documentation

4.40.1.1 SimulatorThread::SimulatorThread ([ThyrixMainFrame](#) * iniFrame)

Definition at line 6 of file SimulatorThread.cpp.

References [setFramesPerSecond\(\)](#), and [setTimeFactor\(\)](#).

4.40.1.2 SimulatorThread::~[~SimulatorThread](#) () [virtual]

Definition at line 16 of file SimulatorThread.cpp.

4.40.2 Member Function Documentation

4.40.2.1 void * SimulatorThread::Entry () [virtual]

Definition at line 60 of file SimulatorThread.cpp.

References World::advanceTime(), expectedTime, frame, frameInterval, World::getDt(), ThyrixMainFrame::paint(), simulationTimer, sleepIfAhead(), and world.

4.40.2.2 void SimulatorThread::OnExit () [virtual]

Called when the thread exits - whether it terminates normally or is stopped with Delete() (but not when it is Kill()ed!)

Definition at line 90 of file SimulatorThread.cpp.

4.40.2.3 void SimulatorThread::setFramesPerSecond (int *fps*)

Definition at line 36 of file SimulatorThread.cpp.

References frameInterval.

Referenced by ThyrixMainFrame::onSetFps(), and SimulatorThread().

4.40.2.4 void SimulatorThread::setPause ()

Definition at line 25 of file SimulatorThread.cpp.

References paused.

Referenced by ThyrixMainFrame::onSetSpeed().

4.40.2.5 void SimulatorThread::setTimeFactor (float *newFactor*)

Definition at line 29 of file SimulatorThread.cpp.

References expectedTime, paused, simulationTimer, and timeFactor.

Referenced by ThyrixMainFrame::onSetSpeed(), SimulatorThread(), and sleepIfAhead().

4.40.2.6 void SimulatorThread::sleepIfAhead () [private]

Sleeps to adjust the speed of the simulator to a predefined speed.

Definition at line 40 of file SimulatorThread.cpp.

References expectedTime, setTimeFactor(), simulationTimer, and timeFactor.

Referenced by Entry().

4.40.2.7 void SimulatorThread::step ()

Advances the simulator with a timestep. Called directly by the frame, following a command event.

Definition at line 19 of file SimulatorThread.cpp.

References `World::advanceTime()`, and `world`.

Referenced by `ThyrixMainFrame::onStep()`.

4.40.3 Member Data Documentation

4.40.3.1 float `SimulatorThread::expectedTime` [private]

The time (real time, relative to the starting time of the simulator or a later reference time) at which the next simulator timestep should be executed, in s.

Definition at line 54 of file `SimulatorThread.h`.

Referenced by `Entry()`, `setTimeFactor()`, and `sleepIfAhead()`.

4.40.3.2 `ThyrixMainFrame*` `SimulatorThread::frame` [private]

A pointer to the frame on which it will draw.

Definition at line 61 of file `SimulatorThread.h`.

Referenced by `Entry()`.

4.40.3.3 unsigned `SimulatorThread::frameInterval` [private]

The time interval for displaying a frame, in ms.

Definition at line 47 of file `SimulatorThread.h`.

Referenced by `Entry()`, and `setFramesPerSecond()`.

4.40.3.4 bool `SimulatorThread::paused` [private]

Indicates that the simulation is paused.

Definition at line 50 of file `SimulatorThread.h`.

Referenced by `setPause()`, and `setTimeFactor()`.

4.40.3.5 `wxStopWatch` `SimulatorThread::simulationTimer` [private]

A timer used for synchronizing the simulator time at a desired speed factor. Counts the time since the starting of the simulation or a later reference time.

Definition at line 58 of file `SimulatorThread.h`.

Referenced by `Entry()`, `setTimeFactor()`, and `sleepIfAhead()`.

4.40.3.6 float `SimulatorThread::timeFactor` [private]

The time factor between the speed of the simulator and real time.

Definition at line 44 of file `SimulatorThread.h`.

Referenced by `setTimeFactor()`, and `sleepIfAhead()`.

4.40.3.7 [World*](#) [SimulatorThread::world](#)

Definition at line 37 of file SimulatorThread.h.

Referenced by [Entry\(\)](#), and [step\(\)](#).

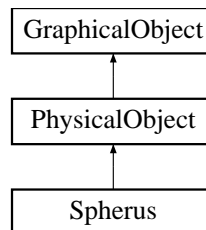
The documentation for this class was generated from the following files:

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

4.41 Spherus Class Reference

```
#include <Spherus.h>
```

Inheritance diagram for Spherus::



Public Member Functions

- [Spherus](#) ([real](#) x, [real](#) y, [real](#) alpha=0.0, [real](#) extension=0.6, [real](#) R=0.1, [Controller](#) *controller=NULL)
- virtual [~Spherus](#) ()
- void [computeState](#) ()
- void [computeSinCos](#) ()
- void [computeMemberPositions](#) ()
- void [computeCircleProprioception](#) ([Circle](#) *circle, float &forward, float &backward)
- virtual void [computeDerivativesWithoutContacts](#) ([ContactSolver](#) *contactSolver)
- virtual void [computeDerivatives](#) ([GlobalContactInfoVector](#) *globalContacts)
- virtual void [integrate](#) (const [Integrator](#) &integrator)
- virtual void [controll](#) ()
- virtual bool [detectContacts](#) ([PhysicalObject](#) *object, [GlobalContactInfoVector](#) *contacts)
- virtual void [detectInternalContacts](#) ([GlobalContactInfoVector](#) *contacts)
- virtual void [deleteContacts](#) ()
- virtual bool [detectMouseContact](#) (const [Vector2](#) &rMouse, [Vector2](#) &p, [PhysicalObject](#) *&object)
- void [draw](#) ([GUI](#) *gui)

Public Attributes

- [real](#) R
- [real](#) extension
- [Circle](#) * circles [2]
- [VisualSensor](#) * eyes [2]
- [real](#) rocketActivations [2]
- [real](#) rocketForceValues [2]
- [Vector2](#) rocketForces [2]
- [real](#) extensionActivation
- [real](#) targetExtension
- [real](#) sinAlpha
- [real](#) cosAlpha
- [Controller](#) * controller

Static Public Attributes

- const unsigned int `nSensors` = $2 * (nVisionSensors + nTactileSensors) + 1 + 4$
- const unsigned int `nEffectors` = $2 * 2 + 1$

4.41.1 Detailed Description

Defines the Spherus agent. The agent is composed of two circles, each at distance l from the center r of the agent. The angle of the line defined by the circles with the horizontal is α .

Definition at line 22 of file Spherus.h.

4.41.2 Constructor & Destructor Documentation

4.41.2.1 Spherus::Spherus (real x , real y , real $\alpha = 0.0$, real $extension = 0.6$, real $R = 0.1$, `Controller` * $controller = \text{NULL}$)

Definition at line 19 of file Spherus.cpp.

References `circles`, `computeMemberPositions()`, `computeSinCos()`, `extensionActivation`, `eyes`, `M_PI`, `nTactileSensors`, `nVisionSensors`, `rocketActivations`, `rocketForceValues`, `Vector2::setXY()`, and `targetExtension`.

4.41.2.2 Spherus::~Spherus () [virtual]

Definition at line 46 of file Spherus.cpp.

References `circles`, and `eyes`.

4.41.3 Member Function Documentation

4.41.3.1 void Spherus::computeCircleProprioception (`Circle` * $circle$, float & $forward$, float & $backward$)

Definition at line 133 of file Spherus.cpp.

References `PhysicalObject::m`, `M_PI`, `real`, `PhysicalObject::v`, `Vector2::x`, and `Vector2::y`.

Referenced by `controll()`.

4.41.3.2 void Spherus::computeDerivatives (`GlobalContactInfoVector` * $globalContacts$) [virtual]

Computes the derivatives, given the contact forces. The `globalContacts` pointer is needed for contact processing for articulated agents.

Reimplemented from `PhysicalObject`.

Definition at line 201 of file Spherus.cpp.

References `circles`, and `PhysicalObject::computeDerivatives()`.

4.41.3.3 void Spherus::computeDerivativesWithoutContacts ([ContactSolver](#) * *contactSolver*) [virtual]

Computes the derivatives (velocity, angular velocity) of the object, ignoring any eventual contact forces.

Reimplemented from [PhysicalObject](#).

Definition at line 147 of file Spherus.cpp.

References [circles](#), [PhysicalObject::computeDerivativesWithoutContacts\(\)](#), [cosAlpha](#), [elasticK](#), [extension](#), [PhysicalObject::externalForce](#), [real](#), [rocketForces](#), [rocketForceValues](#), [Vector2::setXY\(\)](#), and [sinAlpha](#).

4.41.3.4 void Spherus::computeMemberPositions ()

Definition at line 71 of file Spherus.cpp.

References [circles](#), [cosAlpha](#), [extension](#), [PhysicalObject::r](#), [Vector2::setXY\(\)](#), [sinAlpha](#), [Vector2::x](#), and [Vector2::y](#).

Referenced by [Spherus\(\)](#).

4.41.3.5 void Spherus::computeSinCos ()

Definition at line 66 of file Spherus.cpp.

References [cosAlpha](#), and [sinAlpha](#).

Referenced by [computeState\(\)](#), and [Spherus\(\)](#).

4.41.3.6 void Spherus::computeState ()

Computes alpha and extension as a function of the positions of the two eyes, calls [computeSinCos](#).

Definition at line 54 of file Spherus.cpp.

References [circles](#), [computeSinCos\(\)](#), [extension](#), [PhysicalObject::r](#), [real](#), [sqr\(\)](#), [Vector2::x](#), and [Vector2::y](#).

Referenced by [integrate\(\)](#).

4.41.3.7 void Spherus::controll () [virtual]

Transmits perceptual information to the controller and gets motor information from it. To be overridden by agents or other objects with self-generated movement.

Reimplemented from [PhysicalObject](#).

Definition at line 76 of file Spherus.cpp.

References [Controller::advanceTime\(\)](#), [circles](#), [computeCircleProprioception\(\)](#), [controller](#), [extension](#), [extensionActivation](#), [eyes](#), [Controller::getOutput\(\)](#), [maxExtension](#), [rocketActivations](#), [rocketForceValues](#), [Controller::setInput\(\)](#), and [targetExtension](#).

4.41.3.8 void Spherus::deleteContacts () [virtual]

Delete the contacts of the object.

Reimplemented from [PhysicalObject](#).

Definition at line 186 of file Spherus.cpp.

References circles, `PhysicalObject::deleteContacts()`, and eyes.

4.41.3.9 `bool Spherus::detectContacts (PhysicalObject * object, GlobalContactInfoVector * contacts)` [virtual]

Contact handling.

Implements [PhysicalObject](#).

Definition at line 173 of file Spherus.cpp.

References circles, `VisualSensor::detectContacts()`, `Circle::detectContacts()`, and eyes.

4.41.3.10 `void Spherus::detectInternalContacts (GlobalContactInfoVector * contacts)` [virtual]

Detects internal contacts. Does nothing for primitive objects or composed objects. To be overridden by complex objects that have parts that move relative to each other, such as articulated objects.

Reimplemented from [PhysicalObject](#).

Definition at line 182 of file Spherus.cpp.

References circles, and `Circle::detectContacts()`.

4.41.3.11 `bool Spherus::detectMouseContact (const Vector2 & rMouse, Vector2 & p, PhysicalObject *& object)` [virtual]

Detects whether a click of the mouse at `rMouse` has touched the object. In case of contact, returns true and sets `p` to the relative vector between the center of the object and the contact point; `p` is expressed in the object reference frame because it rotates with the object while the object is dragged. The object needs to be set by the function because, for articulated objects, the dragged object is an articulation and not the whole articulated object. To be overridden by the various object primitives.

Reimplemented from [PhysicalObject](#).

Definition at line 193 of file Spherus.cpp.

References circles, and `Circle::detectMouseContact()`.

4.41.3.12 `void Spherus::draw (GUI * gui)` [virtual]

Draw the object to the graphical user interface.

Implements [GraphicalObject](#).

Definition at line 219 of file Spherus.cpp.

References circles, `Circle::draw()`, `VisualSensor::draw()`, `GUI::drawForce()`, `GUI::drawLine()`, eyes, `PhysicalObject::r`, `rocketForces`, `GUI::setBrushColor()`, `GUI::setPenColor()`, `Vector2::x`, and `Vector2::y`.

4.41.3.13 `void Spherus::integrate (const Integrator & integrator)` [virtual]

Advances the time to the next timestep.

Reimplemented from [PhysicalObject](#).

Definition at line 208 of file Spherus.cpp.

References [PhysicalObject::alpha](#), [circles](#), [computeState\(\)](#), and [PhysicalObject::integrate\(\)](#).

4.41.4 Member Data Documentation

4.41.4.1 [Circle*](#) [Spherus::circles](#)[2]

The circles composing the agent

Definition at line 35 of file Spherus.h.

Referenced by [computeDerivatives\(\)](#), [computeDerivativesWithoutContacts\(\)](#), [computeMemberPositions\(\)](#), [computeState\(\)](#), [controll\(\)](#), [deleteContacts\(\)](#), [detectContacts\(\)](#), [detectInternalContacts\(\)](#), [detectMouseContact\(\)](#), [draw\(\)](#), [integrate\(\)](#), [Spherus\(\)](#), and [~Spherus\(\)](#).

4.41.4.2 [Controller*](#) [Spherus::controller](#)

Definition at line 75 of file Spherus.h.

Referenced by [controll\(\)](#).

4.41.4.3 [real](#) [Spherus::cosAlpha](#)

Definition at line 51 of file Spherus.h.

Referenced by [computeDerivativesWithoutContacts\(\)](#), [computeMemberPositions\(\)](#), and [computeSinCos\(\)](#).

4.41.4.4 [real](#) [Spherus::extension](#)

The distance between the circles.

Definition at line 32 of file Spherus.h.

Referenced by [computeDerivativesWithoutContacts\(\)](#), [computeMemberPositions\(\)](#), [computeState\(\)](#), and [controll\(\)](#).

4.41.4.5 [real](#) [Spherus::extensionActivation](#)

Definition at line 42 of file Spherus.h.

Referenced by [controll\(\)](#), and [Spherus\(\)](#).

4.41.4.6 [VisualSensor*](#) [Spherus::eyes](#)[2]

Definition at line 37 of file Spherus.h.

Referenced by [controll\(\)](#), [deleteContacts\(\)](#), [detectContacts\(\)](#), [draw\(\)](#), [Spherus\(\)](#), and [~Spherus\(\)](#).

4.41.4.7 `const unsigned int Spherus::nEffectors = 2*2+1` [static]

Definition at line 17 of file Spherus.cpp.

4.41.4.8 `const unsigned int Spherus::nSensors = 2*(nVisionSensors+nTactileSensors)+1+4`
[static]

The number of unit sensors (pixels) in the tactile sensor.

Reimplemented from [PhysicalObject](#).

Definition at line 16 of file Spherus.cpp.

4.41.4.9 `real Spherus::R`

The radius of the circles composing the agent

Definition at line 29 of file Spherus.h.

4.41.4.10 `real Spherus::rocketActivations[2]`

Definition at line 39 of file Spherus.h.

Referenced by `controll()`, and `Spherus()`.

4.41.4.11 `Vector2 Spherus::rocketForces[2]`

Definition at line 41 of file Spherus.h.

Referenced by `computeDerivativesWithoutContacts()`, and `draw()`.

4.41.4.12 `real Spherus::rocketForceValues[2]`

Definition at line 40 of file Spherus.h.

Referenced by `computeDerivativesWithoutContacts()`, `controll()`, and `Spherus()`.

4.41.4.13 `real Spherus::sinAlpha`

Definition at line 51 of file Spherus.h.

Referenced by `computeDerivativesWithoutContacts()`, `computeMemberPositions()`, and `computeSin-Cos()`.

4.41.4.14 `real Spherus::targetExtension`

The target distance between the circles

Definition at line 45 of file Spherus.h.

Referenced by `controll()`, and `Spherus()`.

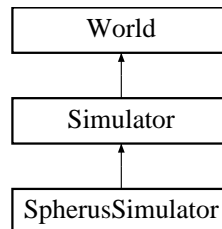
The documentation for this class was generated from the following files:

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

4.42 SpherusSimulator Class Reference

```
#include <SpherusSimulator.h>
```

Inheritance diagram for SpherusSimulator::



Public Member Functions

- [SpherusSimulator\(\)](#)
- virtual [~SpherusSimulator\(\)](#)

Public Attributes

- [Spherus * agent](#)

4.42.1 Constructor & Destructor Documentation

4.42.1.1 SpherusSimulator::SpherusSimulator()

r, x, y.

Definition at line 7 of file SpherusSimulator.cpp.

References [agent](#), [degrees](#), [real](#), and [Simulator::registerObject\(\)](#).

4.42.1.2 SpherusSimulator::~~SpherusSimulator() [virtual]

Definition at line 40 of file SpherusSimulator.cpp.

4.42.2 Member Data Documentation

4.42.2.1 Spherus* SpherusSimulator::agent

Definition at line 17 of file SpherusSimulator.h.

Referenced by [SpherusSimulator\(\)](#).

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

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

4.43 SpikeGraph Class Reference

```
#include <SpikeGraph.h>
```

Public Member Functions

- [SpikeGraph](#) (int [nNeurons](#), int [bufferSize](#))
- virtual [~SpikeGraph](#) ()
- void [push](#) (int neuron, bool value)
- void [advanceTime](#) ()
- bool [getValue](#) (int neuron, int i)
- void [draw](#) (wxDC *outdc, int x, int y, int h=1)

Public Attributes

- int [bufferSize](#)
- int [dataSize](#)
- int [nNeurons](#)
- int [index](#)

Private Attributes

- bool ** [data](#)

4.43.1 Detailed Description

Draws a spikegram (used to illustrate the activity of a spiking neural network).

Definition at line 12 of file SpikeGraph.h.

4.43.2 Constructor & Destructor Documentation

4.43.2.1 SpikeGraph::SpikeGraph (int *nNeurons*, int *bufferSize*)

Definition at line 11 of file SpikeGraph.cpp.

References [data](#), [dataSize](#), and [index](#).

4.43.2.2 SpikeGraph::~SpikeGraph () [virtual]

Definition at line 20 of file SpikeGraph.cpp.

References [data](#).

4.43.3 Member Function Documentation

4.43.3.1 void SpikeGraph::advanceTime () [inline]

Definition at line 40 of file SpikeGraph.h.

4.43.3.2 void SpikeGraph::draw (wxDC * *outdc*, int *x*, int *y*, int *h* = 1)

Definition at line 27 of file SpikeGraph.cpp.

References `getValue()`.

4.43.3.3 bool SpikeGraph::getValue (int *neuron*, int *i*) `[inline]`

Definition at line 48 of file SpikeGraph.h.

Referenced by `draw()`.

4.43.3.4 void SpikeGraph::push (int *neuron*, bool *value*) `[inline]`

Definition at line 31 of file SpikeGraph.h.

4.43.4 Member Data Documentation**4.43.4.1 int SpikeGraph::bufferSize**

The size of the memory buffer of the graph

Definition at line 18 of file SpikeGraph.h.

4.43.4.2 bool SpikeGraph::data** `[private]`

Definition at line 57 of file SpikeGraph.h.

Referenced by `SpikeGraph()`, and `~SpikeGraph()`.

4.43.4.3 int SpikeGraph::dataSize

The actual size of the data that was already set.

Definition at line 21 of file SpikeGraph.h.

Referenced by `SpikeGraph()`.

4.43.4.4 int SpikeGraph::index

The index that marks the start of the data. Virtually, the new data values added to the graph are pushed at the end of the data stack, while old values are popped at the beginning. In practice, just the index that marks the start of the data moves, while the new data values are overwritten over the older ones.

Definition at line 29 of file SpikeGraph.h.

Referenced by `SpikeGraph()`.

4.43.4.5 int SpikeGraph::nNeurons

Definition at line 23 of file SpikeGraph.h.

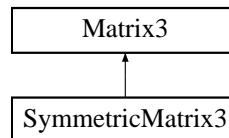
The documentation for this class was generated from the following files:

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

4.44 SymmetricMatrix3 Class Reference

```
#include <SymmetricMatrix3.h>
```

Inheritance diagram for SymmetricMatrix3::



Public Member Functions

- [SymmetricMatrix3 \(\)](#)
- virtual [~SymmetricMatrix3 \(\)](#)
- void [setElements \(real v1, real v2, real v3, real v4, real v5, real v6\)](#)
- [real getDeterminant \(\) const](#)
- void [mirror \(\)](#)
- [SymmetricMatrix3 & operator+= \(const SymmetricMatrix3 &m\)](#)

4.44.1 Detailed Description

A class for symmetric 3x3 matrices. Only the upper half of the matrix should be used. The data storage is still a matrix, for convenient access.

Definition at line 13 of file SymmetricMatrix3.h.

4.44.2 Constructor & Destructor Documentation

4.44.2.1 SymmetricMatrix3::SymmetricMatrix3 () [inline]

Definition at line 15 of file SymmetricMatrix3.h.

4.44.2.2 virtual SymmetricMatrix3::~~SymmetricMatrix3 () [inline, virtual]

Definition at line 16 of file SymmetricMatrix3.h.

4.44.3 Member Function Documentation

4.44.3.1 [real SymmetricMatrix3::getDeterminant \(\) const](#) [inline]

Gets the determinant of the matrix, based on the upper half elements only.

Definition at line 28 of file SymmetricMatrix3.h.

References [real](#), and [sqr\(\)](#).

Referenced by [ArticulatedAgentQuasistatic::backwardDynamics\(\)](#), and [ArticulatedAgentBase::solveSystem\(\)](#).

4.44.3.2 void SymmetricMatrix3::mirror () [inline]

Sets the lower half of the matrix to correspond to the upper half.

Definition at line 37 of file SymmetricMatrix3.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), and ArticulatedAgentBase::solveSystem().

4.44.3.3 [SymmetricMatrix3](#)& SymmetricMatrix3::operator+= (const [SymmetricMatrix3](#) & m) [inline]

The + operator is restricted to the upper half of the matrix.

Definition at line 44 of file SymmetricMatrix3.h.

References Matrix3::matrix.

4.44.3.4 void SymmetricMatrix3::setElements (real v1, real v2, real v3, real v4, real v5, real v6) [inline]

Sets the elements from the upper half of the matrix.

Definition at line 19 of file SymmetricMatrix3.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics().

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

- [SymmetricMatrix3.h](#)

4.45 SystemSolver Class Reference

```
#include <SystemSolver.h>
```

Public Member Functions

- [SystemSolver](#) ()
- virtual [~SystemSolver](#) ()

Static Public Member Functions

- void [luDecompose](#) ([real](#) **a, int n, int *index, [real](#) *tempVector)
- void [luSubstitute](#) ([real](#) **a, int n, int *index, [real](#) *b)
- void [svDecompose](#) ([real](#) **a, int n, [real](#) *w, [real](#) **v, [real](#) *tempVector)
- void [svSubstitute](#) ([real](#) **u, int n, [real](#) *w, [real](#) **v, [real](#) *b, [real](#) *tempVector)

4.45.1 Detailed Description

Functions for solving linear systems of equations. After Numerical Recipes (Press et al.)

Definition at line 12 of file SystemSolver.h.

4.45.2 Constructor & Destructor Documentation

4.45.2.1 SystemSolver::SystemSolver ()

Definition at line 12 of file SystemSolver.cpp.

4.45.2.2 SystemSolver::~SystemSolver () [virtual]

Definition at line 15 of file SystemSolver.cpp.

4.45.3 Member Function Documentation

4.45.3.1 void SystemSolver::luDecompose ([real](#) ** a, int n, int * index, [real](#) * tempVector) [static]

Given a square matrix a of dimension n, the routine replaces it by the LU decomposition of a rowwise permutation of itself. The original a matrix is destroyed and replaced with the output. index[0..n-1] is an output vector that records the row permutation. tempVector is a temporary vector, it may be preallocated to a size at least n.

Definition at line 18 of file SystemSolver.cpp.

References [real](#).

4.45.3.2 void SystemSolver::luSubstitute (real ** a, int n, int * index, real * b) [static]

Solves the set of equations $ax=b$. a should previously be decomposed with luDecompose. index is the vector provided by luDecompose. b is destroyed and replaced with the solution vector.

Definition at line 66 of file SystemSolver.cpp.

References real.

4.45.3.3 void SystemSolver::svDecompose (real ** a, int n, real * w, real ** v, real * tempVector) [static]

Singular value decomposition of square matrix a of dimension n as $u w v^T$. The data in the original matrix is destroyed, and is replaced with u.

Definition at line 86 of file SystemSolver.cpp.

References MathTools::max(), MathTools::modulus(), real, and MathTools::setSign().

Referenced by ContactSolver::driveToZero().

4.45.3.4 void SystemSolver::svSubstitute (real ** u, int n, real * w, real ** v, real * b, real * tempVector) [static]

Solves the set of equations $u w v^T x=b$. u, w, v should previously be generated with svDecompose. w is replaced with the solution vector.

Definition at line 262 of file SystemSolver.cpp.

References real.

Referenced by ContactSolver::driveToZero().

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

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

4.46 ThyrixApplication Class Reference

```
#include <ThyrixApplication.h>
```

Public Member Functions

- [ThyrixApplication \(\)](#)
- virtual [~ThyrixApplication \(\)](#)

Public Attributes

- [World *](#) world

4.46.1 Constructor & Destructor Documentation

4.46.1.1 ThyrixApplication::ThyrixApplication ()

Definition at line 7 of file ThyrixApplication.cpp.

4.46.1.2 ThyrixApplication::~~ThyrixApplication () [virtual]

Definition at line 10 of file ThyrixApplication.cpp.

4.46.2 Member Data Documentation

4.46.2.1 [World*](#) ThyrixApplication::world

Definition at line 19 of file ThyrixApplication.h.

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

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

4.47 ThyrixMainFrame Class Reference

```
#include <ThyrixMainFrame.h>
```

Public Types

- enum {
 [Unused](#) = 100, [kMenuTimePause](#), [kMenuTimeDiv10](#), [kMenuTimeX1](#),
 [kMenuTimeX3](#), [kMenuTimeX10](#), [kMenuTimeX30](#), [kMenuTimeX100](#),
 [kMenuTimeMax](#), [kMenuFps30](#), [kMenuFps60](#), [kMenuFps90](#),
 [kMenuStep](#), [kMenuCapture](#) }

Public Member Functions

- [ThyrixMainFrame](#) (const wxString &title, [World](#) *world, int [bufferWidth](#)=600, int [bufferHeight](#)=400, int [zoom](#)=100)
- virtual [~ThyrixMainFrame](#) ()
- void [start](#) ()
- void [buildMenu](#) ()
- void [OnClose](#) (wxCloseEvent &event)
- void [onSetSpeed](#) (wxCommandEvent &event)
- void [onSetFps](#) (wxCommandEvent &event)
- void [onMotion](#) (wxMouseEvent &event)
- void [onLeftDown](#) (wxMouseEvent &event)
- void [onLeftUp](#) (wxMouseEvent &event)
- void [onRightDown](#) (wxMouseEvent &event)
- void [onRightUp](#) (wxMouseEvent &event)
- void [onStep](#) (wxCommandEvent &event)
- void [onCapture](#) (wxCommandEvent &event)
- void [paintDC](#) (wxDC *dc)
- void [paint](#) ()

Public Attributes

- [World](#) * [world](#)

Private Member Functions

- [DECLARE_CLASS](#) ([ThyrixMainFrame](#))
- [DECLARE_EVENT_TABLE](#) ()

Private Attributes

- [SimulatorThread](#) `thread`
- `wxMemoryDC` `buffer`
- `wxBitmap` `bitmap`
- `int` `bufferWidth`
- `int` `bufferHeight`
- `GUIWx` * `gui`
- `int` `zoom`

4.47.1 Member Enumeration Documentation

4.47.1.1 anonymous enum

Enumeration values:

Unused

kMenuTimePause

kMenuTimeDiv10

kMenuTimeX1

kMenuTimeX3

kMenuTimeX10

kMenuTimeX30

kMenuTimeX100

kMenuTimeMax

kMenuFps30

kMenuFps60

kMenuFps90

kMenuStep

kMenuCapture

Definition at line 12 of file `ThyrixMainFrame.h`.

4.47.2 Constructor & Destructor Documentation

4.47.2.1 `ThyrixMainFrame::ThyrixMainFrame (const wxString & title, World * world, int bufferWidth = 600, int bufferHeight = 400, int zoom = 100)`

4.47.2.2 `ThyrixMainFrame::~ThyrixMainFrame ()` [virtual]

Definition at line 90 of file `ThyrixMainFrame.cpp`.

References `gui`, and `thread`.

4.47.3 Member Function Documentation

4.47.3.1 void ThyrixMainFrame::buildMenu ()

Definition at line 96 of file ThyrixMainFrame.cpp.

References kMenuFps30, kMenuFps60, kMenuFps90, kMenuTimeDiv10, kMenuTimeMax, kMenuTimePause, kMenuTimeX1, kMenuTimeX10, kMenuTimeX100, kMenuTimeX3, and kMenuTimeX30.

4.47.3.2 ThyrixMainFrame::DECLARE_CLASS ([ThyrixMainFrame](#)) [private]

4.47.3.3 ThyrixMainFrame::DECLARE_EVENT_TABLE () [private]

4.47.3.4 void ThyrixMainFrame::onCapture (wxCommandEvent & *event*)

Definition at line 130 of file ThyrixMainFrame.cpp.

References bufferHeight, bufferWidth, and paintDC().

4.47.3.5 void ThyrixMainFrame::OnClose (wxCloseEvent & *event*)

Definition at line 196 of file ThyrixMainFrame.cpp.

4.47.3.6 void ThyrixMainFrame::onLeftDown (wxMouseEvent & *event*)

Captures pressing of mouse left button.

Definition at line 249 of file ThyrixMainFrame.cpp.

References gui, GUI::inverseMapX(), GUI::inverseMapY(), World::onMouseLeftDown(), and world.

4.47.3.7 void ThyrixMainFrame::onLeftUp (wxMouseEvent & *event*)

Captures release of mouse left button.

Definition at line 257 of file ThyrixMainFrame.cpp.

References gui, GUI::inverseMapX(), GUI::inverseMapY(), World::onMouseLeftUp(), and world.

4.47.3.8 void ThyrixMainFrame::onMotion (wxMouseEvent & *event*)

Captures mouse motion.

Definition at line 243 of file ThyrixMainFrame.cpp.

References gui, GUI::inverseMapX(), GUI::inverseMapY(), World::setMouseCoordinates(), and world.

4.47.3.9 void ThyrixMainFrame::onRightDown (wxMouseEvent & *event*)

Definition at line 264 of file ThyrixMainFrame.cpp.

References gui, GUI::inverseMapX(), GUI::inverseMapY(), World::onMouseRightDown(), and world.

4.47.3.10 void ThyrixMainFrame::onRightUp (wxMouseEvent & event)

Definition at line 270 of file ThyrixMainFrame.cpp.

References `gui`, `GUI::inverseMapX()`, `GUI::inverseMapY()`, `World::onMouseRightUp()`, and `world`.

4.47.3.11 void ThyrixMainFrame::onSetFps (wxCommandEvent & event)

Definition at line 140 of file ThyrixMainFrame.cpp.

References `kMenuFps30`, `kMenuFps60`, `kMenuFps90`, `SimulatorThread::setFramesPerSecond()`, and `thread`.

4.47.3.12 void ThyrixMainFrame::onSetSpeed (wxCommandEvent & event)

Definition at line 154 of file ThyrixMainFrame.cpp.

References `kMenuTimeDiv10`, `kMenuTimeMax`, `kMenuTimePause`, `kMenuTimeX1`, `kMenuTimeX10`, `kMenuTimeX100`, `kMenuTimeX3`, `kMenuTimeX30`, `SimulatorThread::setPause()`, `SimulatorThread::setTimeFactor()`, and `thread`.

4.47.3.13 void ThyrixMainFrame::onStep (wxCommandEvent & event)

Definition at line 126 of file ThyrixMainFrame.cpp.

References `SimulatorThread::step()`, and `thread`.

4.47.3.14 void ThyrixMainFrame::paint ()

Paints the image to the memory buffer and then to the screen.

Definition at line 229 of file ThyrixMainFrame.cpp.

References `buffer`, `bufferHeight`, `bufferWidth`, and `paintDC()`.

Referenced by `SimulatorThread::Entry()`.

4.47.3.15 void ThyrixMainFrame::paintDC (wxDC * dc)

Paints the simulator generated image to the specified device context.

Definition at line 277 of file ThyrixMainFrame.cpp.

References `World::draw()`, `gui`, `GUIWx::setDC()`, and `world`.

Referenced by `onCapture()`, and `paint()`.

4.47.3.16 void ThyrixMainFrame::start ()

Creates the [GUI](#) object (needed by the simulator) and starts the thread.

Definition at line 121 of file ThyrixMainFrame.cpp.

References `bufferHeight`, `gui`, `thread`, and `zoom`.

4.47.4 Member Data Documentation

4.47.4.1 wxBitmap [ThyrixMainFrame::bitmap](#) [private]

Bitmap used for graphical buffering.

Definition at line 101 of file ThyrixMainFrame.h.

4.47.4.2 wxMemoryDC [ThyrixMainFrame::buffer](#) [private]

Memory device context used as a graphical buffer. It provides a means to draw on a bitmap.

Definition at line 98 of file ThyrixMainFrame.h.

Referenced by `paint()`.

4.47.4.3 int [ThyrixMainFrame::bufferHeight](#) [private]

The dimensions of the graphical buffer.

Definition at line 104 of file ThyrixMainFrame.h.

Referenced by `onCapture()`, `paint()`, and `start()`.

4.47.4.4 int [ThyrixMainFrame::bufferWidth](#) [private]

The dimensions of the graphical buffer.

Definition at line 104 of file ThyrixMainFrame.h.

Referenced by `onCapture()`, and `paint()`.

4.47.4.5 GUIWx* [ThyrixMainFrame::gui](#) [private]

The [GUI](#) object (an interface to the [GUI](#) which is available to the simulator).

Definition at line 107 of file ThyrixMainFrame.h.

Referenced by `onLeftDown()`, `onLeftUp()`, `onMotion()`, `onRightDown()`, `onRightUp()`, `paintDC()`, `start()`, and `~ThyrixMainFrame()`.

4.47.4.6 SimulatorThread [ThyrixMainFrame::thread](#) [private]

Definition at line 95 of file ThyrixMainFrame.h.

Referenced by `onSetFps()`, `onSetSpeed()`, `onStep()`, `start()`, and `~ThyrixMainFrame()`.

4.47.4.7 World* [ThyrixMainFrame::world](#)

Definition at line 83 of file ThyrixMainFrame.h.

Referenced by `onLeftDown()`, `onLeftUp()`, `onMotion()`, `onRightDown()`, `onRightUp()`, and `paintDC()`.

4.47.4.8 int [ThyrixMainFrame::zoom](#) [private]

The zoom applied by the [GUI](#) object.

Definition at line 110 of file ThyrixMainFrame.h.

Referenced by [start\(\)](#).

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

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

4.48 ThyrixParameters Class Reference

```
#include <ThyrixParameters.h>
```

Public Member Functions

- [ThyrixParameters](#) ()
- virtual [~ThyrixParameters](#) ()

Static Public Attributes

- const [real](#) [epsilonContact](#) = ([real](#))0.002
- const [real](#) [halfEpsilonContact](#) = [ThyrixParameters::epsilonContact](#)/2
- const [real](#) [defaultDensity](#) = ([real](#))20.0
- const [real](#) [kContact](#) = [ThyrixParameters::epsilonContact](#)*[ThyrixParameters::defaultDensity](#)*1000
- const [real](#) [infinity](#) = ([real](#))1e7
- [real](#) [kMouseForce](#) = 5

4.48.1 Detailed Description

Defines some basic parameters for the simulator.

Definition at line 30 of file [ThyrixParameters.h](#).

4.48.2 Constructor & Destructor Documentation

4.48.2.1 [ThyrixParameters::ThyrixParameters](#) ()

Definition at line 19 of file [ThyrixParameters.cpp](#).

4.48.2.2 [ThyrixParameters::~~ThyrixParameters](#) () [virtual]

Definition at line 23 of file [ThyrixParameters.cpp](#).

4.48.3 Member Data Documentation

4.48.3.1 const [real](#) [ThyrixParameters::defaultDensity](#) = ([real](#))20.0 [static]

The default surface density of objects in the simulator.

Definition at line 10 of file [ThyrixParameters.cpp](#).

4.48.3.2 const [real](#) [ThyrixParameters::epsilonContact](#) = ([real](#))0.002 [static]

Tolerance length for estimating contact at contact detection.

Definition at line 7 of file [ThyrixParameters.cpp](#).

4.48.3.3 `const real ThyrixParameters::halfEpsilonContact = ThyrixParameters::epsilonContact/2`
[static]

1/2 epsilonContact.

Definition at line 8 of file ThyrixParameters.cpp.

4.48.3.4 `const real ThyrixParameters::infinity = (real)1e7` [static]

A number larger than the values of any coordinates of objects in the simulator.

Definition at line 14 of file ThyrixParameters.cpp.

4.48.3.5 `const real ThyrixParameters::kContact = ThyrixParameters::epsilonContact*ThyrixParameters::defaultDensity*1000` [static]

The elastic constant used for penalty forces in case of penetration.

Definition at line 12 of file ThyrixParameters.cpp.

4.48.3.6 `real ThyrixParameters::kMouseForce = 5` [static]

An elastic constant used for computing the force generated by dragging an object with the mouse.

Definition at line 16 of file ThyrixParameters.cpp.

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

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

4.49 Vector2 Class Reference

```
#include <Vector2.h>
```

Public Member Functions

- [Vector2](#) ()
- [Vector2](#) (real x, real y)
- virtual [~Vector2](#) ()
- void [setToZero](#) ()
- void [setXY](#) (real x, real y)
- void [setElement](#) (int i, real value)
- real [getModule](#) ()
- real [getSquaredModule](#) ()
- void [normalize](#) ()
- void [updateMin](#) (const [Vector2](#) &r)
- void [updateMax](#) (const [Vector2](#) &r)
- void [rotate](#) (real alpha)
- void [cross](#) (real w)
- real [operator\[\]](#) (int i)
- [Vector2](#) [operator+](#) (const [Vector2](#) &v) const
- [Vector2](#) [operator-](#) (const [Vector2](#) &v) const
- [Vector2](#) [operator *](#) (const real r) const
- [Vector2](#) & [operator+=](#) (const [Vector2](#) &v)
- [Vector2](#) & [operator-=](#) (const [Vector2](#) &v)
- [Vector2](#) & [operator+=](#) (const real r)
- [Vector2](#) & [operator-=](#) (const real r)
- [Vector2](#) & [operator *=](#) (const real r)
- [Vector2](#) & [operator/=](#) (const real r)
- real [operator *](#) ([Vector2](#) v)
- real [operator^](#) ([Vector2](#) v)

Public Attributes

- [real](#) x
- [real](#) y

4.49.1 Detailed Description

A 2D vector.

Definition at line 11 of file Vector2.h.

4.49.2 Constructor & Destructor Documentation

4.49.2.1 [Vector2::Vector2](#) () [inline]

The default constructor, sets the vector to 0.

Definition at line 16 of file Vector2.h.

4.49.2.2 Vector2::Vector2 (real x, real y) [inline]

Constructor.

Definition at line 19 of file Vector2.h.

4.49.2.3 virtual Vector2::~~Vector2 () [inline, virtual]

Destructor; does nothing.

Definition at line 22 of file Vector2.h.

4.49.3 Member Function Documentation

4.49.3.1 void Vector2::cross (real w) [inline]

The vector r becomes $w \times r$ (cross product), w being a vector having just a z axis component.

Definition at line 78 of file Vector2.h.

References real.

Referenced by `PhysicalObject::fillContactMatrix()`, `ArticulatedAgentBase::fillContactMatrix()`, and `ArticulatedAgentQuasistatic::forwardAccelerations()`.

4.49.3.2 real Vector2::getModule () [inline]

Gets the module of the vector: $\sqrt{x^2+y^2}$.

Definition at line 37 of file Vector2.h.

References real, and `sqr()`.

Referenced by `ElasticLink::applyForces()`, `CappedRectangle::detectContacts()`, and `CappedRectangle::detectMouseContact()`.

4.49.3.3 real Vector2::getSquaredModule () [inline]

Gets the squared module of the vector: x^2+y^2 .

Definition at line 44 of file Vector2.h.

References real.

Referenced by `ComposedPhysicalObject::computeMassProperties()`, `CappedRectangle::detectContacts()`, and `Circle::detectMouseContact()`.

4.49.3.4 void Vector2::normalize () [inline]

Normalizes the vector to have unit module

Definition at line 47 of file Vector2.h.

References real.

Referenced by `Pac::controll()`, `Circle::detectContacts()`, and `CappedRectangle::detectContacts()`.

4.49.3.5 `real Vector2::operator * (Vector2 v)` [inline]

The scalar product.

Definition at line 112 of file Vector2.h.

References `real`, `x`, and `y`.

4.49.3.6 `Vector2 Vector2::operator * (const real r) const` [inline]

Definition at line 101 of file Vector2.h.

4.49.3.7 `Vector2& Vector2::operator *= (const real r)` [inline]

Definition at line 109 of file Vector2.h.

4.49.3.8 `Vector2 Vector2::operator+ (const Vector2 & v) const` [inline]

Definition at line 93 of file Vector2.h.

References `x`, and `y`.

4.49.3.9 `Vector2& Vector2::operator+= (const real r)` [inline]

Definition at line 107 of file Vector2.h.

4.49.3.10 `Vector2& Vector2::operator+= (const Vector2 & v)` [inline]

Definition at line 105 of file Vector2.h.

References `x`, and `y`.

4.49.3.11 `Vector2 Vector2::operator- (const Vector2 & v) const` [inline]

Definition at line 97 of file Vector2.h.

References `x`, and `y`.

4.49.3.12 `Vector2& Vector2::operator-= (const real r)` [inline]

Definition at line 108 of file Vector2.h.

4.49.3.13 `Vector2& Vector2::operator-= (const Vector2 & v)` [inline]

Definition at line 106 of file Vector2.h.

References `x`, and `y`.

4.49.3.14 `Vector2& Vector2::operator/= (const real r)` [inline]

Definition at line 110 of file Vector2.h.

4.49.3.15 `]`

`real` Vector2::operator[] (int *i*) `[inline]`

Gets element *i*; *i* is numbered starting from 0 (*i*=0,1).

Definition at line 86 of file Vector2.h.

References `real`.

4.49.3.16 `real` Vector2::operator^ (Vector2 *v*) `[inline]`

The vector product (having a component only on the z axis; returns this component).

Definition at line 114 of file Vector2.h.

References `real`, `x`, and `y`.

4.49.3.17 `void` Vector2::rotate (`real` *alpha*) `[inline]`

Rotates the vector with angle *alpha*. If the vector is initially expressed in a reference system rotated with *alpha* relative to the laboratory reference system, after rotation the vector will be expressed in the laboratory reference system.

Definition at line 69 of file Vector2.h.

References `real`.

Referenced by `Simulator::applyMouseForce()`, `ArticulatedAgentQuasistatic::backwardDynamics()`, `ArticulatedAgentQuasistatic::computeBodyDerivatives()`, `ArticulatedAgentQuasistatic::computeBodyDerivativesWithoutContacts()`, `ArticulatedLink::computeForceQuasistatic()`, `ComposedPhysicalObject::computeMemberPositions()`, `Pac::controll()`, `ComposedPhysicalObject::detectMouseContact()`, `Circle::detectMouseContact()`, `CappedRectangle::detectMouseContact()`, `ArticulatedAgentBase::fillContactMatrix()`, `ArticulatedAgentQuasistatic::forwardAccelerations()`, and `ArticulatedAgentQuasistatic::forwardKinematics()`.

4.49.3.18 `void` Vector2::setElement (int *i*, `real` *value*) `[inline]`

Sets element *i* to value *value*; *i* is numbered starting from 1 (*i*=1,2).

Definition at line 34 of file Vector2.h.

4.49.3.19 `void` Vector2::setToZero () `[inline]`

Sets the vector to 0.

Definition at line 28 of file Vector2.h.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, `PhysicalObject::computeDerivativesWithoutContacts()`, `ArticulatedLink::computeForceQuasistatic()`, `ComposedPhysicalObject::computeMassProperties()`, `ArticulatedLink::deleteContacts()`, `Circle::detectContacts()`, and `CappedRectangle::detectContacts()`.

4.49.3.20 `void` Vector2::setXY (`real` *x*, `real` *y*) `[inline]`

Sets *x* and *y* components.

Definition at line 31 of file Vector2.h.

Referenced by `ComposedPhysicalObject::addObject()`, `Iunctus::build()`, `ComposedPhysicalObject::ComposedPhysicalObject()`, `ArticulatedAgentQuasistatic::computeBodyDerivatives()`, `ArticulatedAgentQuasistatic::computeBodyDerivativesWithoutContacts()`, `Spherus::computeDerivativesWithoutContacts()`, `Spherus::computeMemberPositions()`, `ComposedPhysicalObject::computeMemberPositions()`, `CappedRectangle::detectContacts()`, `CappedRectangle::detectMouseContact()`, `ArticulatedAgentBase::fillContactMatrix()`, `ArticulatedAgentQuasistatic::forwardAccelerations()`, `ArticulatedAgentQuasistatic::forwardKinematics()`, `Elastoid::integrate()`, `PhysicalObject::PhysicalObject()`, `Spherus::Spherus()`, and `VisualSensor::VisualSensor()`.

4.49.3.21 void Vector2::updateMax (const Vector2 & r) [inline]

Set vector components to the maximum of either current components or r components.

Definition at line 60 of file Vector2.h.

References `x`, and `y`.

Referenced by `ComposedPhysicalObject::computeMemberPositions()`, `Elastoid::integrate()`, `ArticulatedAgentQuasistatic::integrate()`, and `ArticulatedAgentQuasistatic::rollback()`.

4.49.3.22 void Vector2::updateMin (const Vector2 & r) [inline]

Set vector components to the minimum of either current components or r components.

Definition at line 54 of file Vector2.h.

References `x`, and `y`.

Referenced by `ComposedPhysicalObject::computeMemberPositions()`, `Elastoid::integrate()`, `ArticulatedAgentQuasistatic::integrate()`, and `ArticulatedAgentQuasistatic::rollback()`.

4.49.4 Member Data Documentation

4.49.4.1 real Vector2::x

Definition at line 24 of file Vector2.h.

Referenced by `Simulator::applyMouseForce()`, `ArticulatedAgentQuasistatic::backwardDynamics()`, `Border::Border()`, `CappedRectangle::CappedRectangle()`, `Circle::Circle()`, `ArticulatedAgentQuasistatic::computeBodyDerivatives()`, `Spherus::computeCircleProprioception()`, `ArticulatedLink::computeForceQuasistatic()`, `ArticulatedComponent::computeIStar0()`, `Spherus::computeMemberPositions()`, `Spherus::computeState()`, `ArticulatedAgentQuasistatic::computeTotalForces()`, `VisualSensor::detectContacts()`, `Circle::detectContacts()`, `CappedRectangle::detectContacts()`, `CappedRectangle::detectMouseContact()`, `VisualSensor::draw()`, `Spherus::draw()`, `Iunctus::draw()`, `Circle::draw()`, `CappedRectangle::draw()`, `ArticulatedLink::draw()`, `GUI::drawCircle()`, `GUIWx::drawForce()`, `GUI::drawLine()`, `Circle::drawSensors()`, `CappedRectangle::drawSensors()`, `GUIWx::drawTorque()`, `ArticulatedAgentQuasistatic::forwardAccelerations()`, `Circle::getISensor()`, `CappedRectangle::getISensorSemicircle()`, `CappedRectangle::getLateralSide()`, `Integrator::integrate()`, `operator*()`, `operator+()`, `operator+=()`, `operator-()`, `operator-=()`, `operator^()`, `PhysicalObject::setPosition()`, `PhysicalObject::setVelocity()`, `updateMax()`, and `updateMin()`.

4.49.4.2 [real Vector2::y](#)

Definition at line 25 of file Vector2.h.

Referenced by Simulator::applyMouseForce(), ArticulatedAgentQuasistatic::backwardDynamics(), Border::Border(), CappedRectangle::CappedRectangle(), Circle::Circle(), Spherus::computeCircleProprioception(), ArticulatedLink::computeForceQuasistatic(), ArticulatedComponent::computeIStar0(), Spherus::computeMemberPositions(), Spherus::computeState(), ArticulatedAgentQuasistatic::computeTotalForces(), VisualSensor::detectContacts(), Circle::detectContacts(), CappedRectangle::detectContacts(), CappedRectangle::detectMouseContact(), VisualSensor::draw(), Spherus::draw(), Iunctus::draw(), Circle::draw(), CappedRectangle::draw(), ArticulatedLink::draw(), GUI::drawCircle(), GUIWx::drawForce(), GUI::drawLine(), Circle::drawSensors(), CappedRectangle::drawSensors(), GUIWx::drawTorque(), ArticulatedAgentQuasistatic::forwardAccelerations(), Circle::getISensor(), CappedRectangle::getISensorSemicircle(), CappedRectangle::getLateralSide(), Integrator::integrate(), operator *(), operator +(), operator +=(), operator -, operator -=(), operator ^(), PhysicalObject::setPosition(), PhysicalObject::setVelocity(), updateMax(), and updateMin().

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

- [Vector2.h](#)

4.50 Vector3 Class Reference

```
#include <Vector3.h>
```

Public Member Functions

- [Vector3](#) ()
- [Vector3](#) (real x, real y, real z)
- virtual [~Vector3](#) ()
- void [setToZero](#) ()
- void [setXYZ](#) (real x, real y, real z)
- void [setElement](#) (int i, real value)
- real [getModule](#) ()
- real [getSquaredModule](#) ()
- void [rotate](#) (real alpha)
- real [operator\[\]](#) (int i)
- [Vector3](#) & [operator+=](#) (const [Vector3](#) &v)
- [Vector3](#) & [operator-=](#) (const [Vector3](#) &v)
- [Vector3](#) & [operator *=](#) (const real r)
- [Vector3](#) & [operator/=](#) (const real r)

Public Attributes

- real x
- real y
- real z

4.50.1 Detailed Description

A 3D vector.

Definition at line 13 of file Vector3.h.

4.50.2 Constructor & Destructor Documentation

4.50.2.1 [Vector3::Vector3](#) () [inline]

The default constructor, sets the vector to 0.

Definition at line 16 of file Vector3.h.

4.50.2.2 [Vector3::Vector3](#) (real x, real y, real z) [inline]

Constructor.

Definition at line 18 of file Vector3.h.

4.50.2.3 `virtual Vector3::~~Vector3 () [inline, virtual]`

Destructor; does nothing.

Definition at line 20 of file Vector3.h.

4.50.3 Member Function Documentation

4.50.3.1 `real Vector3::getModule () [inline]`

Gets the module of the vector: $\sqrt{x^2+y^2+z^2}$.

Definition at line 44 of file Vector3.h.

References `real`.

4.50.3.2 `real Vector3::getSquaredModule () [inline]`

Gets the squared module of the vector: $x^2+y^2+z^2$.

Definition at line 47 of file Vector3.h.

References `real`.

4.50.3.3 `Vector3& Vector3::operator *= (const real r) [inline]`

Definition at line 74 of file Vector3.h.

4.50.3.4 `Vector3& Vector3::operator+= (const Vector3 & v) [inline]`

Definition at line 72 of file Vector3.h.

References `x`, `y`, and `z`.

4.50.3.5 `Vector3& Vector3::operator-= (const Vector3 & v) [inline]`

Definition at line 73 of file Vector3.h.

References `x`, `y`, and `z`.

4.50.3.6 `Vector3& Vector3::operator/= (const real r) [inline]`

Definition at line 75 of file Vector3.h.

4.50.3.7 `[]`

`real Vector3::operator[] (int i) [inline]`

Gets element `i`; `i` is numbered starting from 0 (`i=0,1,2`).

Definition at line 63 of file Vector3.h.

References `real`.

4.50.3.8 void Vector3::rotate (real *alpha*) [inline]

Rotates the spatial part of the vector with angle *alpha*. If the vector is initially expressed in a reference system rotated with *alpha* relative to the laboratory reference system, after rotation the vector will be expressed in the laboratory reference system.

Definition at line 53 of file Vector3.h.

References `real`.

4.50.3.9 void Vector3::setElement (int *i*, real *value*) [inline]

Sets element *i* to value *value*; *i* is numbered starting from 1 (*i*=1,2,3).

Definition at line 33 of file Vector3.h.

Referenced by `ArticulatedAgentBase::solveSystem()`.

4.50.3.10 void Vector3::setToZero () [inline]

Sets the vector to 0.

Definition at line 27 of file Vector3.h.

4.50.3.11 void Vector3::setXYZ (real *x*, real *y*, real *z*) [inline]

Sets *x*, *y* and *z* components.

Definition at line 30 of file Vector3.h.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, `ArticulatedAgentQuasistatic::forwardAccelerations()`, and `ArticulatedAgentQuasistatic::computeBodyDerivativesWithoutContacts()`.

4.50.4 Member Data Documentation**4.50.4.1 real Vector3::x**

Definition at line 22 of file Vector3.h.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, `ArticulatedAgentQuasistatic::computeBodyDerivativesWithoutContacts()`, `ArticulatedLink::computeForceQuasistatic()`, `Iunctus::controll()`, `ArticulatedAgentQuasistatic::forwardAccelerations()`, `operator+=()`, `operator-=()`, and `Matrix3::setColumn()`.

4.50.4.2 real Vector3::y

Definition at line 23 of file Vector3.h.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, `ArticulatedAgentQuasistatic::computeBodyDerivativesWithoutContacts()`, `ArticulatedLink::computeForceQuasistatic()`, `Iunctus::controll()`, `ArticulatedAgentQuasistatic::forwardAccelerations()`, `operator+=()`, `operator-=()`, and `Matrix3::setColumn()`.

4.50.4.3 [real Vector3::z](#)

Definition at line 24 of file Vector3.h.

Referenced by `ArticulatedAgentQuasistatic::backwardDynamics()`, `ArticulatedAgentQuasistatic::computeBodyDerivativesWithoutContacts()`, `ArticulatedLink::computeForceQuasistatic()`, `Iunctus::controll()`, `ArticulatedAgentQuasistatic::forwardAccelerations()`, `operator+=()`, `operator-=()`, and `Matrix3::setColumn()`.

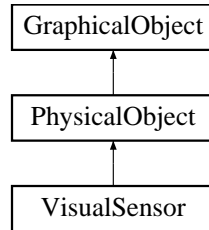
The documentation for this class was generated from the following file:

- [Vector3.h](#)

4.51 VisualSensor Class Reference

```
#include <VisualSensor.h>
```

Inheritance diagram for VisualSensor::



Public Member Functions

- [VisualSensor](#) ([PhysicalObject](#) *parent, [real](#) relativeX, [real](#) relativeY, [real](#) relativeAlpha, [real](#) R, [real](#) viewAngle, unsigned int nSensors, std::string label="")
- virtual [~VisualSensor](#) ()
- virtual bool [detectContacts](#) ([PhysicalObject](#) *object, [GlobalContactInfoVector](#) *contacts)
- bool [detectContacts](#) ([Border](#) *border, [GlobalContactInfoVector](#) *contacts)
- bool [detectContacts](#) ([Circle](#) *circle, [GlobalContactInfoVector](#) *contacts)
- bool [detectContacts](#) ([CappedRectangle](#) *capsule, [GlobalContactInfoVector](#) *contacts)
- virtual void [draw](#) ([GUI](#) *gui)

Private Member Functions

- void [activate](#) ([real](#) gammaM, [real](#) gammaP)
- [real](#) [computeGamma](#) ([real](#) dx, [real](#) dy)

Private Attributes

- [real](#) R
- [real](#) viewAngle
- [real](#) delta
- [real](#) epsilon

4.51.1 Detailed Description

A visual sensor, composed of unit sensors (pixels) arranged on a circular segment. It currently has infinite range. The sensor is attached to an object, at relative position relativeR and relative angle relativeAlpha. The view angle is centered around the orientation of the sensor.

Definition at line 12 of file VisualSensor.h.

4.51.2 Constructor & Destructor Documentation

4.51.2.1 VisualSensor::VisualSensor ([PhysicalObject](#) * *parent*, [real](#) *relativeX*, [real](#) *relativeY*, [real](#) *relativeAlpha*, [real](#) *R*, [real](#) *viewAngle*, unsigned int *nSensors*, std::string *label* = "")

Constructor.

Parameters:

parent is the object it is attached to, at relative coordinates

relativeX and

relativeY and relative angle. The view angle should be between 0..2Pi.

Definition at line 7 of file VisualSensor.cpp.

References [delta](#), [epsilon](#), [M_PI](#), and [Vector2::setXY\(\)](#).

4.51.2.2 VisualSensor::~~VisualSensor () [virtual]

Definition at line 24 of file VisualSensor.cpp.

4.51.3 Member Function Documentation

4.51.3.1 void VisualSensor::activate ([real](#) *gammaM*, [real](#) *gammaP*) [private]

Activates all sensors between gammaM and gammaP. The interval [gammaM, gammaP] is considered to be fully occluded by an object.

Definition at line 107 of file VisualSensor.cpp.

References [epsilon](#), and [real](#).

Referenced by [detectContacts\(\)](#).

4.51.3.2 [real](#) VisualSensor::computeGamma ([real](#) *dx*, [real](#) *dy*) [inline, private]

Computes the angle gamma corresponding to a point at dx, dy relative to the sensor center. The value is normalized between (-Pi,Pi].

Definition at line 62 of file VisualSensor.h.

References [PhysicalObject::alpha](#), and [real](#).

Referenced by [detectContacts\(\)](#).

4.51.3.3 bool VisualSensor::detectContacts ([CappedRectangle](#) * *capsule*, [GlobalContactInfoVector](#) * *contacts*) [virtual]

Tests for contacts between this object and a capsule.

Reimplemented from [PhysicalObject](#).

Definition at line 48 of file VisualSensor.cpp.

References [activate\(\)](#), [computeGamma\(\)](#), [CappedRectangle::cosAlpha](#), [CappedRectangle::l](#), [MathTools::modulus\(\)](#), [CappedRectangle::R](#), [PhysicalObject::r](#), [real](#), [CappedRectangle::sinAlpha](#), [Vector2::x](#), and [Vector2::y](#).

4.51.3.4 **bool VisualSensor::detectContacts** ([Circle](#) * *circle*, [GlobalContactInfoVector](#) * *contacts*) [virtual]

Tests for contacts between this object and a circle.

Reimplemented from [PhysicalObject](#).

Definition at line 27 of file VisualSensor.cpp.

References [activate\(\)](#), [computeGamma\(\)](#), [MathTools::modulus\(\)](#), [Circle::R](#), [PhysicalObject::r](#), [real](#), [Vector2::x](#), and [Vector2::y](#).

4.51.3.5 **bool VisualSensor::detectContacts** ([Border](#) * *border*, [GlobalContactInfoVector](#) * *contacts*) [inline, virtual]

Tests for contacts between this object and a border.

Reimplemented from [PhysicalObject](#).

Definition at line 29 of file VisualSensor.h.

4.51.3.6 **virtual bool VisualSensor::detectContacts** ([PhysicalObject](#) * *object*, [GlobalContactInfoVector](#) * *contacts*) [inline, virtual]

Redirects contact detection.

Implements [PhysicalObject](#).

Definition at line 24 of file VisualSensor.h.

References [PhysicalObject::detectContacts\(\)](#).

Referenced by [Spherus::detectContacts\(\)](#), and [IunctusSimulator::detectContacts\(\)](#).

4.51.3.7 **void VisualSensor::draw** ([GUI](#) * *gui*) [virtual]

Draw the object to the graphical user interface.

Implements [GraphicalObject](#).

Definition at line 178 of file VisualSensor.cpp.

References [PhysicalObject::alpha](#), [delta](#), [GUI::drawLine\(\)](#), [PhysicalObject::r](#), [real](#), [GUI::setPenColor\(\)](#), [Vector2::x](#), and [Vector2::y](#).

Referenced by [Spherus::draw\(\)](#).

4.51.4 Member Data Documentation

4.51.4.1 **real VisualSensor::delta** [private]

Half the view angle.

Definition at line 51 of file VisualSensor.h.

Referenced by [draw\(\)](#), and [VisualSensor\(\)](#).

4.51.4.2 [real VisualSensor::epsilon](#) [private]

The view angle of an unit sensor (pixel).

Definition at line 54 of file VisualSensor.h.

Referenced by activate(), and VisualSensor().

4.51.4.3 [real VisualSensor::R](#) [private]

The radius of the sensor.

Definition at line 45 of file VisualSensor.h.

4.51.4.4 [real VisualSensor::viewAngle](#) [private]

The view angle of the sensor.

Definition at line 48 of file VisualSensor.h.

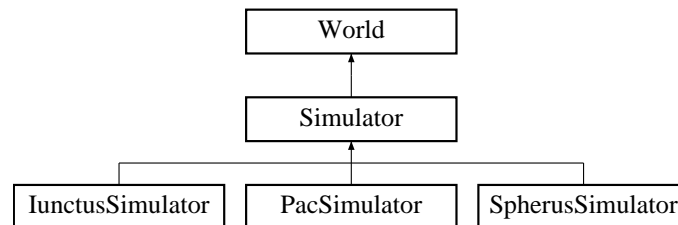
The documentation for this class was generated from the following files:

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

4.52 World Class Reference

```
#include <World.h>
```

Inheritance diagram for World::



Public Member Functions

- [World \(\)](#)
- virtual [~World \(\)](#)
- virtual double [getDt \(\)](#) const
- virtual void [advanceTime \(\)](#)=0
- virtual void [draw \(GUI *gui\)](#)
- virtual void [setMouseCoordinates](#) (float x, float y)
- virtual void [onMouseLeftDown](#) (float x, float y)
- virtual void [onMouseRightDown](#) (float x, float y)
- virtual void [onMouseLeftUp](#) (float x, float y)
- virtual void [onMouseRightUp](#) (float x, float y)

Public Attributes

- float [mouseX](#)
- float [mouseY](#)
- bool [isRightMouseButtonPressed](#)
- bool [isLeftMouseButtonPressed](#)
- unsigned long int [timeStep](#)

4.52.1 Detailed Description

A world is a dynamic system that is updated in discrete time and can be graphically illustrated. This class provides just an interface, and implements interaction through mouse events.

Definition at line 14 of file World.h.

4.52.2 Constructor & Destructor Documentation

4.52.2.1 World::World ()

Definition at line 12 of file World.cpp.

4.52.2.2 `World::~~World ()` [virtual]

Definition at line 15 of file `World.cpp`.

4.52.3 Member Function Documentation

4.52.3.1 `virtual void World::advanceTime ()` [pure virtual]

Implemented in [Simulator](#).

Referenced by `SimulatorThread::Entry()`, and `SimulatorThread::step()`.

4.52.3.2 `void World::draw (GUI * gui)` [virtual]

Reimplemented in [Simulator](#).

Definition at line 18 of file `World.cpp`.

References `GUI::outTextStatusBar()`, and `timeStep`.

Referenced by `Simulator::draw()`, and `ThyrixMainFrame::paintDC()`.

4.52.3.3 `virtual double World::getDt () const` [inline, virtual]

Reimplemented in [Simulator](#).

Definition at line 19 of file `World.h`.

Referenced by `SimulatorThread::Entry()`.

4.52.3.4 `virtual void World::onMouseLeftDown (float x, float y)` [inline, virtual]

Reimplemented in [Simulator](#).

Definition at line 36 of file `World.h`.

Referenced by `ThyrixMainFrame::onLeftDown()`, and `Simulator::onMouseLeftDown()`.

4.52.3.5 `virtual void World::onMouseLeftUp (float x, float y)` [inline, virtual]

Reimplemented in [Simulator](#).

Definition at line 46 of file `World.h`.

Referenced by `ThyrixMainFrame::onLeftUp()`, and `Simulator::onMouseLeftUp()`.

4.52.3.6 `virtual void World::onMouseRightDown (float x, float y)` [inline, virtual]

Definition at line 41 of file `World.h`.

Referenced by `ThyrixMainFrame::onRightDown()`.

4.52.3.7 virtual void World::onMouseRightUp (float x, float y) [inline, virtual]

Definition at line 51 of file World.h.

Referenced by ThyrixMainFrame::onRightUp().

4.52.3.8 virtual void World::setMouseCoordinates (float x, float y) [inline, virtual]

Definition at line 31 of file World.h.

Referenced by ThyrixMainFrame::onMotion().

4.52.4 Member Data Documentation**4.52.4.1 bool [World::isLeftMouseButtonPressed](#)**

Definition at line 29 of file World.h.

4.52.4.2 bool [World::isRightMouseButtonPressed](#)

This may be used for sending custom events.

Definition at line 28 of file World.h.

4.52.4.3 float [World::mouseX](#)

Coordinates, in world space, of the position of the mouse.

Definition at line 26 of file World.h.

4.52.4.4 float [World::mouseY](#)

Coordinates, in world space, of the position of the mouse.

Definition at line 26 of file World.h.

4.52.4.5 unsigned long int [World::timeStep](#)

Definition at line 56 of file World.h.

Referenced by draw().

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

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

Chapter 5

Thyrix File Documentation

5.1 ArticulatedAgentBase.cpp File Reference

```
#include "ArticulatedAgentBase.h"  
#include "Simulator.h"  
#include "GUI.h"  
#include "purgeContainer.h"
```

5.2 ArticulatedAgentBase.h File Reference

```
#include "ArticulatedComponent.h"  
#include "ArticulatedLimb.h"  
#include "Circle.h"  
#include "CappedRectangle.h"  
#include "Vector3.h"
```

Classes

- class [ArticulatedAgentBase](#)

5.3 ArticulatedAgentQuasistatic.cpp File Reference

```
#include "ArticulatedAgentQuasistatic.h"  
#include "Simulator.h"  
#include "ContactInfo.h"  
#include <algorithm>
```

5.4 ArticulatedAgentQuasistatic.h File Reference

```
#include "ArticulatedAgentBase.h"
```

Classes

- class [ArticulatedAgentQuasistatic](#)

5.5 ArticulatedComponent.cpp File Reference

```
#include "ArticulatedComponent.h"  
#include "purgeContainer.h"
```

5.6 ArticulatedComponent.h File Reference

```
#include "LinkContactInfo.h"  
#include "ComposedPhysicalObject.h"  
#include "SymmetricMatrix3.h"  
#include "Vector3.h"
```

Classes

- class [ArticulatedComponent](#)

5.7 ArticulatedLimb.cpp File Reference

```
#include "ArticulatedLimb.h"  
#include "Simulator.h"  
#include "purgeContainer.h"
```

5.8 ArticulatedLimb.h File Reference

```
#include "ArticulatedLink.h"
```

Classes

- class [ArticulatedLimb](#)

Typedefs

- typedef std::vector< [ArticulatedLimb](#) * > [ArticulatedLimbPVector](#)

5.8.1 Typedef Documentation

5.8.1.1 typedef std::vector<[ArticulatedLimb](#)*> [ArticulatedLimbPVector](#)

Definition at line 56 of file ArticulatedLimb.h.

5.9 ArticulatedLink.cpp File Reference

```
#include "ArticulatedLink.h"  
#include "ArticulatedLimb.h"  
#include "Simulator.h"  
#include "Integrator.h"
```

5.10 ArticulatedLink.h File Reference

```
#include "ArticulatedComponent.h"
```

Classes

- class [ArticulatedLink](#)

Typedefs

- typedef std::vector< [ArticulatedLink](#) * > [ArticulatedLinkPVector](#)

Variables

- const [real](#) [DEFAULT_K](#) = ([real](#))0.01
- const [real](#) [DEFAULT_ETA](#) = ([real](#))0.15

5.10.1 Typedef Documentation

5.10.1.1 typedef std::vector<[ArticulatedLink](#)*> [ArticulatedLinkPVector](#)

Definition at line 18 of file ArticulatedLink.h.

5.10.2 Variable Documentation

5.10.2.1 const [real](#) [DEFAULT_ETA](#) = ([real](#))0.15

Default damping factor of the link articulation

Definition at line 12 of file ArticulatedLink.h.

5.10.2.2 const [real](#) [DEFAULT_K](#) = ([real](#))0.01

Default elastic constant of the link articulation

Definition at line 10 of file ArticulatedLink.h.

5.11 Border.cpp File Reference

```
#include "Border.h"  
#include "Simulator.h"
```

5.12 Border.h File Reference

```
#include "PhysicalObject.h"
```

Classes

- class [Border](#)

5.13 CachingController.cpp File Reference

```
#include "CachingController.h"
```

5.14 CachingController.h File Reference

```
#include "Controller.h"  
#include <memory.h>
```

Classes

- class [CachingController](#)

5.15 CappedRectangle.cpp File Reference

```
#include "CappedRectangle.h"  
#include "Simulator.h"  
#include "Circle.h"  
#include "Border.h"
```

5.16 CappedRectangle.h File Reference

```
#include "PhysicalObject.h"
```

Classes

- class [CappedRectangle](#)

5.17 Circle.cpp File Reference

```
#include "Circle.h"  
#include "Simulator.h"  
#include "ContactInfo.h"  
#include "GlobalContactInfo.h"
```

5.18 Circle.h File Reference

```
#include "PhysicalObject.h"  
#include "Border.h"
```

Classes

- class [Circle](#)

5.19 Color.cpp File Reference

```
#include "Color.h"  
#include "ColorDefinitions.h"
```

5.20 Color.h File Reference

Classes

- class [Color](#)

5.21 ColorDefinitions.cpp File Reference

```
#include "ColorDefinitions.h"
```

5.22 ColorDefinitions.h File Reference

```
#include <map>
```

Classes

- class [ColorDefinitions](#)
- struct [ColorDefinitions::SimpleColor](#)
- struct [ColorDefinitions::ltstr](#)

5.23 ComposedPhysicalObject.cpp File Reference

```
#include "ComposedPhysicalObject.h"  
#include "Simulator.h"  
#include "purgeContainer.h"  
#include <string>
```

5.24 ComposedPhysicalObject.h File Reference

```
#include "PhysicalObject.h"
```

Classes

- class [ComposedPhysicalObject](#)

5.25 ContactInfo.cpp File Reference

```
#include "ContactInfo.h"  
#include "PhysicalObject.h"
```

5.26 ContactInfo.h File Reference

```
#include "Vector2.h"  
#include <vector>
```

Classes

- class [ContactInfo](#)

Typedefs

- typedef std::vector< [ContactInfo](#) * > [ContactInfoPVector](#)

Enumerations

- enum [ContactType](#) { [contactTypeForce](#), [contactTypeForceParallel](#), [contactTypeTorque](#) }

5.26.1 Typedef Documentation

5.26.1.1 typedef std::vector<[ContactInfo](#)*> [ContactInfoPVector](#)

Definition at line 74 of file ContactInfo.h.

5.26.2 Enumeration Type Documentation

5.26.2.1 enum [ContactType](#)

Enumeration values:

contactTypeForce
contactTypeForceParallel
contactTypeTorque

Definition at line 11 of file ContactInfo.h.

5.27 ContactSolver.cpp File Reference

```
#include "ContactSolver.h"  
#include "SystemSolver.h"
```

5.28 ContactSolver.h File Reference

```
#include "ThyrixParameters.h"  
#include "GlobalContactInfo.h"
```

Classes

- class [ContactSolver](#)

Enumerations

- enum [ContactSet](#) { [contactSetC](#), [contactSetNC](#), [contactSetI](#) }

5.28.1 Enumeration Type Documentation

5.28.1.1 enum [ContactSet](#)

Enumeration values:

contactSetC

contactSetNC

contactSetI

Definition at line 7 of file ContactSolver.h.

Referenced by [ContactSolver::init\(\)](#).

5.29 Controller.cpp File Reference

```
#include "Controller.h"
```

5.30 Controller.h File Reference

```
#include <assert.h>
```

Classes

- class [Controller](#)

5.31 ElasticLink.cpp File Reference

```
#include "ElasticLink.h"
```

5.32 ElasticLink.h File Reference

```
#include "PhysicalObject.h"
#include "GUI.h"
```

Classes

- class [ElasticLink](#)

Defines

- #define [AFX_ELASTICLINK_H__CF42E708_6848_4CF4_B4DA_19E78FAA2BAC__INCLUDED_](#)

Typedefs

- typedef std::vector< [ElasticLink](#) > [ElasticLinkVector](#)

5.32.1 Define Documentation

5.32.1.1 #define AFX_ELASTICLINK_H__CF42E708_6848_4CF4_B4DA_19E78FAA2BAC__INCLUDED_

Definition at line 6 of file ElasticLink.h.

5.32.2 Typedef Documentation

5.32.2.1 typedef std::vector<[ElasticLink](#)> [ElasticLinkVector](#)

Definition at line 35 of file ElasticLink.h.

5.33 Elastoid.cpp File Reference

```
#include "Elastoid.h"  
#include "Simulator.h"  
#include "purgeContainer.h"
```

5.34 Elastoid.h File Reference

```
#include "ElasticLink.h"
```

Classes

- class [Elastoid](#)

Defines

- #define [AFX_ELASTOID_H__DCB64B95_CC6D_4280_9E1F_242450B50FAF__INCLUDED_](#)

5.34.1 Define Documentation

5.34.1.1 #define AFX_ELASTOID_H__DCB64B95_CC6D_4280_9E1F_242450B50FAF__INCLUDED_

Definition at line 6 of file Elastoid.h.

5.35 GlobalContactInfo.h File Reference

```
#include "ContactInfo.h"
#include "Vector2.h"
```

Classes

- class [GlobalContactInfo](#)

Typedefs

- typedef std::vector< [GlobalContactInfo](#) > [GlobalContactInfoVector](#)

5.35.1 Typedef Documentation

5.35.1.1 typedef std::vector<[GlobalContactInfo](#)> [GlobalContactInfoVector](#)

Definition at line 34 of file GlobalContactInfo.h.

5.36 Graph.cpp File Reference

```
#include "Graph.h"
```

5.37 Graph.h File Reference

```
#include "GraphData.h"  
#include <wx/dcmemory.h>
```

Classes

- class [Graph](#)

Defines

- #define [AFX_GRAPH_H__6AA46D96_39C9_4C8F_9FCC_3446AE0C8C94__INCLUDED_](#)

5.37.1 Define Documentation

5.37.1.1 #define AFX_GRAPH_H__6AA46D96_39C9_4C8F_9FCC_3446AE0C8C94__INCLUDED_

Definition at line 2 of file Graph.h.

5.38 GraphData.cpp File Reference

```
#include "GraphData.h"
```


5.39 GraphData.h File Reference

```
#include <assert.h>
```

Classes

- class [GraphData](#)

Defines

- #define [AFX_GRAPHDATA_H__035AC771_F046_4ECA_BCDE_E570A1138AE4__-INCLUDED_](#)

5.39.1 Define Documentation

5.39.1.1 #define AFX_GRAPHDATA_H__035AC771_F046_4ECA_BCDE_E570A1138AE4__-INCLUDED_

Definition at line 2 of file GraphData.h.

5.40 GraphicalObject.cpp File Reference

```
#include "GraphicalObject.h"
```

5.41 GraphicalObject.h File Reference

```
#include "GUI.h"
```

Classes

- class [GraphicalObject](#)

5.42 GUI.cpp File Reference

```
#include "GUI.h"
```

5.43 GUI.h File Reference

```
#include "Vector2.h"  
#include "Color.h"
```

Classes

- class [GUI](#)

5.44 GUIWx.cpp File Reference

```
#include "GUIWx.h"
#include "Vector2.h"
#include <wx/textctrl.h>
#include <wx/dcmemory.h>
#include <wx/statusbr.h>
#include <wx/pen.h>
```

Functions

- float [cosSum](#) (float *cos1*, float *sin1*, float *cos2*, float *sin2*)
- float [sinSum](#) (float *cos1*, float *sin1*, float *cos2*, float *sin2*)

5.44.1 Function Documentation

5.44.1.1 float [cosSum](#) (float *cos1*, float *sin1*, float *cos2*, float *sin2*) [inline, static]

Definition at line 37 of file GUIWx.cpp.

Referenced by GUIWx::drawCappedRectangle().

5.44.1.2 float [sinSum](#) (float *cos1*, float *sin1*, float *cos2*, float *sin2*) [inline, static]

Definition at line 41 of file GUIWx.cpp.

Referenced by GUIWx::drawCappedRectangle().

5.45 GUIWx.h File Reference

```
#include "GUI.h"  
#include "wxIncludes.h"  
#include <map>
```

Classes

- class [GUIWx](#)

5.46 Integrator.h File Reference

```
#include "Vector2.h"
```

Classes

- class [Integrator](#)

5.47 Iunctus.cpp File Reference

```
#include "Iunctus.h"  
#include "Simulator.h"  
#include "Circle.h"  
#include "CappedRectangle.h"
```

5.48 Iunctus.h File Reference

```
#include "ArticulatedAgentQuasistatic.h"  
#include "RandomController.h"  
#include "VisualSensor.h"
```

Classes

- class [Iunctus](#)

5.49 IunctusSimulator.cpp File Reference

```
#include "IunctusSimulator.h"  
#include "Border.h"  
#include "Circle.h"
```

5.50 IunctusSimulator.h File Reference

```
#include "Simulator.h"  
#include "Iunctus.h"
```

Classes

- class [IunctusSimulator](#)

5.51 LinkContactInfo.cpp File Reference

```
#include "LinkContactInfo.h"
```

5.52 LinkContactInfo.h File Reference

```
#include "Vector3.h"
#include <vector>
```

Classes

- class [LinkContactInfo](#)

Typedefs

- typedef std::vector< [LinkContactInfo](#) * > [LinkContactInfoPVector](#)
- typedef std::vector< [LinkContactInfo](#) > [LinkContactInfoVector](#)

5.52.1 Typedef Documentation

5.52.1.1 typedef std::vector<[LinkContactInfo](#)*> [LinkContactInfoPVector](#)

Definition at line 68 of file LinkContactInfo.h.

5.52.1.2 typedef std::vector<[LinkContactInfo](#)> [LinkContactInfoVector](#)

Definition at line 69 of file LinkContactInfo.h.

5.53 MathTools.cpp File Reference

```
#include "MathTools.h"
```

5.54 MathTools.h File Reference

```
#include <math.h>
```

Classes

- class [MathTools](#)

Defines

- #define [AFX_MATHTOOLS_H__D0539AA7_8BC3_4A5F_99A3_730FAE12C22E__INCLUDED_](#)

5.54.1 Define Documentation

5.54.1.1 #define AFX_MATHTOOLS_H__D0539AA7_8BC3_4A5F_99A3_730FAE12C22E__INCLUDED_

Definition at line 6 of file MathTools.h.

5.55 Matrix3.cpp File Reference

```
#include "Matrix3.h"
```

5.56 Matrix3.h File Reference

```
#include "Vector3.h"
```

Classes

- class [Matrix3](#)

5.57 MTRandom.cpp File Reference

```
#include "MTRandom.h"  
#include <stdio.h>  
#include <string.h>
```

5.58 MTRandom.h File Reference

Classes

- class [MTRandom](#)

Defines

- `#define AFX_MTRANDOM_H__80B8318A_FECC_423A_9199_FF3E5F3A05D3__INCLUDED_`

Variables

- `const int MTRandomN = 624 MTRandomM = 397`

5.58.1 Define Documentation

5.58.1.1 `#define AFX_MTRANDOM_H__80B8318A_FECC_423A_9199_FF3E5F3A05D3__INCLUDED_`

Definition at line 53 of file MTRandom.h.

5.58.2 Variable Documentation

5.58.2.1 `const int MTRandomN = 624 MTRandomM = 397` `[static]`

Definition at line 56 of file MTRandom.h.

5.59 Pac.cpp File Reference

```
#include "Pac.h"  
#include "Circle.h"
```

5.60 Pac.h File Reference

```
#include "Elastoid.h"  
#include "RandomController.h"
```

Classes

- class [Pac](#)

Defines

- #define [AFX_PAC_H__65EEAA15_5429_4476_885F_AF955C54055A__INCLUDED_](#)

5.60.1 Define Documentation

5.60.1.1 #define AFX_PAC_H__65EEAA15_5429_4476_885F_AF955C54055A__INCLUDED_

Definition at line 6 of file Pac.h.

5.61 PacSimulator.cpp File Reference

```
#include "PacSimulator.h"  
#include "Pac.h"  
#include "Circle.h"  
#include "CappedRectangle.h"
```

5.62 PacSimulator.h File Reference

```
#include "Simulator.h"
```

Classes

- class [PacSimulator](#)

Defines

- #define [AFX_PACSIMULATOR_H__E15DFA83_B781_468B_8DD3_411BBECD1FF7__INCLUDED_](#)

5.62.1 Define Documentation

5.62.1.1 #define AFX_PACSIMULATOR_H__E15DFA83_B781_468B_8DD3_411BBECD1FF7__INCLUDED_

Definition at line 6 of file PacSimulator.h.

5.63 PhysicalObject.cpp File Reference

```
#include "PhysicalObject.h"  
#include "Simulator.h"  
#include "Integrator.h"  
#include "purgeContainer.h"  
#include "ContactInfo.h"  
#include "GlobalContactInfo.h"
```

5.64 PhysicalObject.h File Reference

```
#include "GraphicalObject.h"
#include "ContactInfo.h"
#include "GlobalContactInfo.h"
#include "ContactSolver.h"
#include "ThyrixParameters.h"
#include <string>
```

Classes

- class [PhysicalObject](#)

Typedefs

- typedef std::vector< [PhysicalObject](#) * > [PhysicalObjectPVector](#)

5.64.1 Typedef Documentation

5.64.1.1 typedef std::vector<[PhysicalObject](#)*> [PhysicalObjectPVector](#)

Definition at line 211 of file PhysicalObject.h.

5.65 `purgeContainer.h` File Reference

Functions

- `template<class Container> void purgeContainer (Container &container)`

5.65.1 Function Documentation

5.65.1.1 `template<class Container> void purgeContainer (Container & container)`

Purges a container (e.g. `std::vector`) that contains pointers to objects, by deleting these objects, and emptying the container.

Definition at line 6 of file `purgeContainer.h`.

Referenced by `PhysicalObject::deleteContacts()`, `ArticulatedComponent::deleteK()`, `ArticulatedAgentBase::deleteLimbs()`, `ArticulatedLimb::deleteLinks()`, `Simulator::deleteObjects()`, `ComposedPhysicalObject::deleteObjects()`, and `Elastoid::~Elastoid()`.

5.66 Random.cpp File Reference

```
#include "Random.h"  
#include <time.h>
```

5.67 Random.h File Reference

```
#include <stdlib.h>
```

Classes

- class [Random](#)

5.68 RandomController.cpp File Reference

```
#include "RandomController.h"  
#include "Random.h"
```

5.69 RandomController.h File Reference

```
#include "Controller.h"
```

Classes

- class [RandomController](#)

5.70 resource.h File Reference

Defines

- #define DUMMY 4

5.70.1 Define Documentation

5.70.1.1 #define DUMMY 4

Definition at line 5 of file resource.h.

5.71 Simulator.cpp File Reference

```
#include "Simulator.h"  
#include "Border.h"  
#include "Circle.h"  
#include "SystemSolver.h"  
#include "purgeContainer.h"  
#include <algorithm>
```

5.72 Simulator.h File Reference

```
#include "World.h"
#include "Integrator.h"
#include "GlobalContactInfo.h"
#include "PhysicalObject.h"
#include "ContactSolver.h"
#include <set>
```

Classes

- class [Simulator](#)
- class [Simulator::ObjectPair](#)

5.73 SimulatorThread.cpp File Reference

```
#include "SimulatorThread.h"  
#include "ThyrixApplication.h"  
#include "ThyrixMainFrame.h"  
#include "World.h"
```

5.74 SimulatorThread.h File Reference

```
#include "wxIncludes.h"
```

Classes

- class [SimulatorThread](#)

5.75 Spherus.cpp File Reference

```
#include "Spherus.h"  
#include "RandomController.h"  
#include "Circle.h"
```

Variables

- const int **nVisionSensors** = 7
- const int **nTactileSensors** = 8
- const real **maxExtension** = 0.8
- const real **maxExtensionForce** = 0.5
- const real **maxRocketForce** = 0.5
- const real **elasticK** = **maxExtensionForce**/(**maxExtension**/5)

5.75.1 Variable Documentation

5.75.1.1 const real **elasticK** = **maxExtensionForce**/(**maxExtension**/5) [static]

Definition at line 14 of file Spherus.cpp.

Referenced by Spherus::computeDerivativesWithoutContacts().

5.75.1.2 const real **maxExtension** = 0.8 [static]

Definition at line 11 of file Spherus.cpp.

Referenced by Spherus::controll().

5.75.1.3 const real **maxExtensionForce** = 0.5 [static]

Definition at line 12 of file Spherus.cpp.

5.75.1.4 const real **maxRocketForce** = 0.5 [static]

Definition at line 13 of file Spherus.cpp.

5.75.1.5 const int **nTactileSensors** = 8 [static]

Definition at line 6 of file Spherus.cpp.

Referenced by Spherus::Spherus().

5.75.1.6 const int **nVisionSensors** = 7 [static]

Definition at line 5 of file Spherus.cpp.

Referenced by Spherus::Spherus().

5.76 Spherus.h File Reference

```
#include "PhysicalObject.h"  
#include "VisualSensor.h"
```

Classes

- class [Spherus](#)

5.77 SpherusSimulator.cpp File Reference

```
#include "SpherusSimulator.h"  
#include "Spherus.h"  
#include "Border.h"  
#include "Circle.h"  
#include "CappedRectangle.h"
```

5.78 SpherusSimulator.h File Reference

```
#include "Simulator.h"
```

Classes

- class [SpherusSimulator](#)

5.79 SpikeGraph.cpp File Reference

```
#include "SpikeGraph.h"
```

5.80 SpikeGraph.h File Reference

```
#include "assert.h"
#include <wx/dc.h>
```

Classes

- class [SpikeGraph](#)

Defines

- #define [AFX_SPIKEGRAPH_H__BB61CB4C_B0D5_4444_BFA1_4088183CC669__INCLUDED_](#)

5.80.1 Define Documentation

5.80.1.1 #define AFX_SPIKEGRAPH_H__BB61CB4C_B0D5_4444_BFA1_4088183CC669__INCLUDED_

Definition at line 6 of file SpikeGraph.h.

5.81 SymmetricMatrix3.cpp File Reference

```
#include "SymmetricMatrix3.h"
```

5.82 SymmetricMatrix3.h File Reference

```
#include "Matrix3.h"
```

Classes

- class [SymmetricMatrix3](#)

5.83 SystemSolver.cpp File Reference

```
#include "SystemSolver.h"  
#include "MathTools.h"  
#include <stdlib.h>  
#include <assert.h>
```

Defines

- #define [TINY](#) 1.0e-20;

5.83.1 Define Documentation

5.83.1.1 #define TINY 1.0e-20;

Definition at line 6 of file SystemSolver.cpp.

5.84 SystemSolver.h File Reference

```
#include "ThyrixParameters.h"
```

Classes

- class [SystemSolver](#)

5.85 ThyrixApplication.cpp File Reference

```
#include "wxIncludes.h"  
#include <wx/xrc/xmlres.h>  
#include "ThyrixApplication.h"  
#include "ThyrixMainFrame.h"
```

5.86 ThyrixApplication.h File Reference

```
#include "wxIncludes.h"
```

Classes

- class [ThyrixApplication](#)

5.87 ThyrixMainFrame.cpp File Reference

```
#include "wxIncludes.h"
#include <wx/xrc/xmlres.h>
#include <wx/image.h>
#include <wx/menu.h>
#include <wx/toolbar.h>
#include <wx/metafile.h>
#include <wx/gdicmn.h>
#include "ThyrixApplication.h"
#include "ThyrixMainFrame.h"
#include "World.h"
```

Functions

- [EVT_MENU_RANGE](#) (ThyrixMainFrame::kMenuTimePause, ThyrixMainFrame::kMenuTimeMax, ThyrixMainFrame::onSetSpeed) [EVT_MENU_RANGE\(ThyrixMainFrame](#)

5.87.1 Function Documentation

5.87.1.1 [EVT_MENU_RANGE](#) (ThyrixMainFrame::kMenuTimePause, ThyrixMainFrame::kMenuTimeMax, ThyrixMainFrame::onSetSpeed)

Definition at line 32 of file ThyrixMainFrame.cpp.

5.88 ThyrixMainFrame.h File Reference

```
#include "SimulatorThread.h"  
#include "GUIWx.h"
```

Classes

- class [ThyrixMainFrame](#)

5.89 ThyrixParameters.cpp File Reference

```
#include "ThyrixParameters.h"
```

5.90 ThyrixParameters.h File Reference

```
#include <math.h>
```

Classes

- class [ThyrixParameters](#)

Defines

- `#define AFX_THYRIXPARAMETERS_H_442475F3_9963_4247_8192_1505B7F7E4D5__INCLUDED_`
- `#define M_PI 3.1415926535897932384626433832795028841971693993751058209749445923078164062862`

Typedefs

- `typedef double real`

Functions

- `template<class Number> Number sqr (Number x)`

Variables

- `const real degrees = M_PI/180.0`

5.90.1 Define Documentation

5.90.1.1 `#define AFX_THYRIXPARAMETERS_H_442475F3_9963_4247_8192_1505B7F7E4D5__INCLUDED_`

Definition at line 6 of file ThyrixParameters.h.

5.90.1.2 `#define M_PI 3.1415926535897932384626433832795028841971693993751058209749445923078164062862`

Pi constant.

Definition at line 19 of file ThyrixParameters.h.

Referenced by `Iunctus::build()`, `Spherus::computeCircleProprioception()`, `Pac::controll()`, `Circle::detectContacts()`, `CappedRectangle::detectContacts()`, `Iunctus::draw()`, `GUIWx::drawArrow()`, `GUIWx::drawCappedRectangle()`, `Circle::drawSensors()`, `CappedRectangle::drawSensors()`, `GUIWx::drawTorque()`, `CappedRectangle::getISensorSemicircle()`, `Pac::Pac()`, `Spherus::Spherus()`, and `VisualSensor::VisualSensor()`.

5.90.2 Typedef Documentation

5.90.2.1 typedef double **real**

The basic floating point type used by the simulator for representing physical quantities.

Definition at line 12 of file ThyrixParameters.h.

Referenced by VisualSensor::activate(), ElasticLink::applyForces(), ArticulatedAgentQuasistatic::backwardDynamics(), Iunctus::build(), ArticulatedAgentQuasistatic::computeBodyDerivatives(), Circle::computeBox(), Spherus::computeCircleProprioception(), ContactSolver::computeContactsPreda(), ArticulatedAgentQuasistatic::computeDerivatives(), Spherus::computeDerivativesWithoutContacts(), VisualSensor::computeGamma(), ArticulatedComponent::computeIStar0(), Spherus::computeState(), Pac::controll(), Iunctus::controll(), Vector2::cross(), VisualSensor::detectContacts(), Circle::detectContacts(), CappedRectangle::detectContacts(), CappedRectangle::detectMouseContact(), ArticulatedLink::detectTorqueContact(), VisualSensor::draw(), Iunctus::draw(), ArticulatedLink::draw(), GUIWx::drawForce(), Circle::drawSensors(), CappedRectangle::drawSensors(), GUIWx::drawTorque(), ContactSolver::driveToZero(), ArticulatedAgentQuasistatic::forwardAccelerations(), ArticulatedAgentQuasistatic::forwardKinematics(), SymmetricMatrix3::getDeterminant(), Matrix3::getDeterminant(), Integrator::getDt(), Matrix3::getElement(), Circle::getISensor(), CappedRectangle::getISensorLateral(), CappedRectangle::getISensorSemicircle(), CappedRectangle::getLateralSide(), Vector3::getModule(), Vector2::getModule(), Vector3::getSquaredModule(), Vector2::getSquaredModule(), ContactSolver::init(), IunctusSimulator::IunctusSimulator(), SystemSolver::luDecompose(), SystemSolver::luSubstitute(), ContactSolver::maxStep(), Vector2::normalize(), Vector2::operator*(), Vector3::operator[](), Vector2::operator[](), Matrix3::operator[](), Vector2::operator^(), Pac::Pac(), PacSimulator::PacSimulator(), PhysicalObject::PhysicalObject(), Vector3::rotate(), Vector2::rotate(), CappedRectangle::setISensorLateral(), Circle::setSensor(), CappedRectangle::setSensor(), ArticulatedAgentBase::solveSystem(), SpherusSimulator::SpherusSimulator(), SystemSolver::svDecompose(), and SystemSolver::svSubstitute().

5.90.3 Function Documentation

5.90.3.1 template<class Number> Number sqr (Number x) [inline]

Square function.

Definition at line 15 of file ThyrixParameters.h.

Referenced by ArticulatedAgentQuasistatic::backwardDynamics(), Spherus::computeState(), Circle::detectContacts(), CappedRectangle::detectContacts(), SymmetricMatrix3::getDeterminant(), Vector2::getModule(), and MathTools::modulus().

5.90.4 Variable Documentation

5.90.4.1 const **real** degrees = M_PI/180.0

A degree in radians.

Definition at line 23 of file ThyrixParameters.h.

Referenced by Iunctus::build(), IunctusSimulator::IunctusSimulator(), PacSimulator::PacSimulator(), and SpherusSimulator::SpherusSimulator().

5.91 Vector2.cpp File Reference

```
#include "Vector2.h"
```

5.92 Vector2.h File Reference

```
#include "ThyrixParameters.h"  
#include <assert.h>
```

Classes

- class [Vector2](#)

5.93 Vector3.cpp File Reference

```
#include "Vector3.h"
```


5.94 Vector3.h File Reference

```
#include "ThyrixParameters.h"  
#include <assert.h>
```

Classes

- class [Vector3](#)

5.95 VisualSensor.cpp File Reference

```
#include "VisualSensor.h"  
#include "Circle.h"  
#include "CappedRectangle.h"  
#include "MathTools.h"
```

5.96 VisualSensor.h File Reference

```
#include "PhysicalObject.h"
```

Classes

- class [VisualSensor](#)

5.97 World.cpp File Reference

```
#include "World.h"  
#include <stdio.h>
```

5.98 World.h File Reference

```
#include "GUI.h"
```

Classes

- class [World](#)

Defines

- #define [AFX_WORLD_H__A1A33537_BF98_497E_B4CF_3A2839E9C233__INCLUDED_](#)

Variables

- const [real dtDefault](#) = ([real](#))0.02

5.98.1 Define Documentation

5.98.1.1 #define [AFX_WORLD_H__A1A33537_BF98_497E_B4CF_3A2839E9C233__INCLUDED_](#)

Definition at line 6 of file World.h.

5.98.2 Variable Documentation

5.98.2.1 const [real dtDefault](#) = ([real](#))0.02

Definition at line 10 of file World.h.

Referenced by `Simulator::Simulator()`.

5.99 wxIncludes.h File Reference

```
#include <wx/thread.h>
#include <wx/frame.h>
#include <wx/textctrl.h>
#include <wx/app.h>
#include <wx/dcmemory.h>
#include <wx/bitmap.h>
#include <wx/msgdlg.h>
#include <wx/intl.h>
#include <wx/timer.h>
#include <wx/log.h>
```

Index

- ~ArticulatedAgentBase
 - ArticulatedAgentBase, [10](#)
- ~ArticulatedAgentQuasistatic
 - ArticulatedAgentQuasistatic, [15](#)
- ~ArticulatedComponent
 - ArticulatedComponent, [19](#)
- ~ArticulatedLimb
 - ArticulatedLimb, [22](#)
- ~ArticulatedLink
 - ArticulatedLink, [27](#)
- ~Border
 - Border, [34](#)
- ~CachingController
 - CachingController, [37](#)
- ~CappedRectangle
 - CappedRectangle, [40](#)
- ~Circle
 - Circle, [46](#)
- ~Color
 - Color, [50](#)
- ~ColorDefinitions
 - ColorDefinitions, [52](#)
- ~ComposedPhysicalObject
 - ComposedPhysicalObject, [56](#)
- ~ContactInfo
 - ContactInfo, [62](#)
- ~ContactSolver
 - ContactSolver, [66](#)
- ~Controller
 - Controller, [71](#)
- ~ElasticLink
 - ElasticLink, [73](#)
- ~Elastoid
 - Elastoid, [75](#)
- ~GUI
 - GUI, [89](#)
- ~GUIWx
 - GUIWx, [94](#)
- ~Graph
 - Graph, [81](#)
- ~GraphData
 - GraphData, [84](#)
- ~GraphicalObject
 - GraphicalObject, [86](#)
- ~Iunctus
 - Iunctus, [100](#)
- ~IunctusSimulator
 - IunctusSimulator, [103](#)
- ~LinkContactInfo
 - LinkContactInfo, [105](#)
- ~MTRandom
 - MTRandom, [112](#)
- ~MathTools
 - MathTools, [107](#)
- ~Matrix3
 - Matrix3, [109](#)
- ~Pac
 - Pac, [115](#)
- ~PacSimulator
 - PacSimulator, [118](#)
- ~PhysicalObject
 - PhysicalObject, [120](#)
- ~Random
 - Random, [130](#)
- ~RandomController
 - RandomController, [132](#)
- ~Simulator
 - Simulator, [134](#)
- ~SimulatorThread
 - SimulatorThread, [142](#)
- ~Spherus
 - Spherus, [147](#)
- ~SpherusSimulator
 - SpherusSimulator, [153](#)
- ~SpikeGraph
 - SpikeGraph, [154](#)
- ~SymmetricMatrix3
 - SymmetricMatrix3, [157](#)
- ~SystemSolver
 - SystemSolver, [159](#)
- ~ThyrixApplication
 - ThyrixApplication, [161](#)
- ~ThyrixMainFrame
 - ThyrixMainFrame, [163](#)
- ~ThyrixParameters
 - ThyrixParameters, [168](#)
- ~Vector2
 - Vector2, [171](#)
- ~Vector3
 - Vector3, [176](#)

- ~VisualSensor
 - VisualSensor, [181](#)
- ~World
 - World, [184](#)
- activate
 - VisualSensor, [181](#)
- activations
 - PhysicalObject, [126](#)
- activeLink
 - Pac, [116](#)
- addContact
 - PhysicalObject, [121](#)
- addLimb
 - ArticulatedAgentBase, [10](#)
- addLink
 - ArticulatedLimb, [23](#)
- addObject
 - ComposedPhysicalObject, [56](#)
 - Elastoid, [76](#)
- addOneObjectLink
 - ArticulatedLimb, [23](#)
- advanceTime
 - Controller, [71](#)
 - RandomController, [132](#)
 - Simulator, [134](#)
 - SpikeGraph, [154](#)
 - World, [185](#)
- AFX_ELASTICLINK_H__CF42E708_6848_-4CF4_B4DA_19E78FAA2BAC__-INCLUDED_
 - ElasticLink.h, [218](#)
- AFX_ELASTOID_H__DCB64B95_CC6D_4280_-9E1F_242450B50FAF__INCLUDED_
 - Elastoid.h, [220](#)
- AFX_GRAPH_H__6AA46D96_39C9_4C8F_-9FCC_3446AE0C8C94__INCLUDED_
 - Graph.h, [223](#)
- AFX_GRAPHDATA_H__035AC771_F046_-4ECA_BCDE_E570A1138AE4__-INCLUDED_
 - GraphData.h, [225](#)
- AFX_MATHTOOLS_H__D0539AA7_8BC3_-4A5F_99A3_730FAE12C22E__-INCLUDED_
 - MathTools.h, [240](#)
- AFX_MTRANDOM_H__80B8318A_FECC_-423A_9199_FF3E5F3A05D3__-INCLUDED_
 - MTRandom.h, [244](#)
- AFX_PAC_H__65EEAA15_5429_4476_885F_-AF955C54055A__INCLUDED_
 - Pac.h, [246](#)
- AFX_PACSIMULATOR_H__E15DFA83_B781_-468B_8DD3_411BBECD1FF7__-INCLUDED_
 - PacSimulator.h, [248](#)
- AFX_SPIKEGRAPH_H__BB61CB4C_B0D5_-4444_BFA1_4088183CC669__-INCLUDED_
 - SpikeGraph.h, [266](#)
- AFX_THYRIXPARAMETERS_H__442475F3_-9963_4247_8192_1505B7F7E4D5__-INCLUDED_
 - ThyrixParameters.h, [276](#)
- AFX_WORLD_H__A1A33537_BF98_497E_-B4CF_3A2839E9C233__INCLUDED_
 - World.h, [285](#)
- agent
 - IunctusSimulator, [104](#)
 - SpherusSimulator, [153](#)
- aLocal
 - ArticulatedComponent, [19](#)
- alpha
 - ContactInfo, [62](#)
 - GlobalContactInfo, [79](#)
 - LinkContactInfo, [105](#)
 - PhysicalObject, [126](#)
- alphaOld
 - PhysicalObject, [126](#)
- applyForces
 - ElasticLink, [73](#)
- applyMouseForce
 - Simulator, [134](#)
- ArticulatedAgentBase, [9](#)
 - ArticulatedAgentBase, [10](#)
- ArticulatedAgentBase
 - ~ArticulatedAgentBase, [10](#)
 - addLimb, [10](#)
 - ArticulatedAgentBase, [10](#)
 - deleteContacts, [11](#)
 - deleteLimbs, [11](#)
 - detectContacts, [11](#)
 - detectInternalContacts, [11](#)
 - detectMouseContact, [11](#)
 - draw, [12](#)
 - fillContactMatrix, [12](#)
 - limbs, [13](#)
 - registerPrimitives, [12](#)
 - setFillColor, [12](#)
 - setOutlineColor, [13](#)
 - solveSystem, [13](#)
- ArticulatedAgentBase.cpp, [187](#)
- ArticulatedAgentBase.h, [188](#)
- ArticulatedAgentQuasistatic, [14](#)
 - ArticulatedAgentQuasistatic, [15](#)
- ArticulatedAgentQuasistatic

- ~ArticulatedAgentQuasistatic, 15
- ArticulatedAgentQuasistatic, 15
- backwardDynamics, 15
- computeBodyDerivatives, 15
- computeBodyDerivativesWithoutContacts, 15
- computeDerivatives, 16
- computeDerivativesWithoutContacts, 16
- computeForces, 16
- computeTotalForces, 16
- forwardAccelerations, 16
- forwardKinematics, 17
- integrate, 17
- rollback, 17
- ArticulatedAgentQuasistatic.cpp, 189
- ArticulatedAgentQuasistatic.h, 190
- ArticulatedComponent, 18
 - ArticulatedComponent, 19
- ArticulatedComponent
 - ~ArticulatedComponent, 19
 - aLocal, 19
 - ArticulatedComponent, 19
 - beta, 19
 - betaExternal, 19
 - betaStar, 20
 - computeIStar0, 19
 - deleteK, 19
 - IStar, 20
 - IStar0, 20
 - K, 20
 - nContacts, 20
 - Q, 20
 - vLocal, 20
- ArticulatedComponent.cpp, 191
- ArticulatedComponent.h, 192
- ArticulatedLimb, 22
 - ArticulatedLimb, 22
- ArticulatedLimb
 - ~ArticulatedLimb, 22
 - addLink, 23
 - addOneObjectLink, 23
 - ArticulatedLimb, 22
 - computeSinCos, 23
 - cos2, 24
 - cosTheta, 24
 - deleteLinks, 23
 - draw, 23
 - l, 24
 - label, 24
 - links, 24
 - setFillColor, 23
 - setOutlineColor, 23
 - setProperties, 24
 - sin2, 25
 - sinCos, 25
 - sinTheta, 25
 - theta, 25
- ArticulatedLimb.cpp, 193
- ArticulatedLimb.h, 194
- ArticulatedLimb.h
 - ArticulatedLimbPVector, 194
- ArticulatedLimbPVector
 - ArticulatedLimb.h, 194
- ArticulatedLink, 26
 - ArticulatedLink, 27
- ArticulatedLink
 - ~ArticulatedLink, 27
 - ArticulatedLink, 27
 - childLinks, 29
 - computeForceQuasistatic, 27
 - computeMotorTorque, 27
 - computeSinCos, 28
 - coriolis, 29
 - cos2, 29
 - cosTheta, 30
 - deleteContacts, 28
 - deltaThetaMax, 30
 - detectTorqueContact, 28
 - draw, 28
 - eta, 30
 - fillContactMatrix, 28
 - force, 30
 - forceLocal, 30
 - hasElasticTorque, 30
 - integrate, 28
 - k, 31
 - l, 31
 - motorTorque, 31
 - normalizeTheta, 29
 - parentLink, 31
 - rollback, 29
 - setParentLink, 29
 - sin2, 31
 - sinCos, 31
 - sinTheta, 31
 - theta, 32
 - theta0, 32
 - theta0Max, 32
 - theta0Min, 32
 - thetaD, 32
 - thetaDD, 32
 - thetaOld, 33
 - torqueContact, 33
 - totalForce, 33
 - totalTorque, 33
- ArticulatedLink.cpp, 195
- ArticulatedLink.h, 196
- ArticulatedLink.h
 - ArticulatedLinkPVector, 196

- DEFAULT_ETA, 196
 - DEFAULT_K, 196
- ArticulatedLinkPVector
 - ArticulatedLink.h, 196
- b
 - Color, 51
 - ColorDefinitions::SimpleColor, 54
- backwardDynamics
 - ArticulatedAgentQuasistatic, 15
- beta
 - ArticulatedComponent, 19
- betaExternal
 - ArticulatedComponent, 19
- betaStar
 - ArticulatedComponent, 20
- bitmap
 - Graph, 82
 - ThyrixMainFrame, 166
- Border, 34
 - ~Border, 34
 - Border, 34
 - computeDerivativesWithoutContacts, 35
 - detectContacts, 35
 - draw, 35
 - fillContactMatrix, 35
 - integrate, 35
 - normal, 36
 - rollback, 36
- Border.cpp, 197
- Border.h, 198
- boxMax
 - PhysicalObject, 126
- boxMin
 - PhysicalObject, 127
- buffer
 - ThyrixMainFrame, 166
- bufferHeight
 - ThyrixMainFrame, 166
- bufferSize
 - Graph, 82
 - GraphData, 85
 - SpikeGraph, 155
- bufferWidth
 - ThyrixMainFrame, 166
- build
 - Iunctus, 101
- buildMenu
 - ThyrixMainFrame, 164
- cacheInput
 - CachingController, 37
- CachingController, 37
 - CachingController, 37
- CachingController
 - ~CachingController, 37
 - cacheInput, 37
 - CachingController, 37
 - oldInput, 38
- CachingController.cpp, 199
- CachingController.h, 200
- CappedRectangle, 39
 - CappedRectangle, 40
- CappedRectangle
 - ~CappedRectangle, 40
 - CappedRectangle, 40
 - computeBox, 40
 - computeInertia, 40
 - computeMass, 40
 - cosAlpha, 43
 - detectContacts, 41
 - detectMouseContact, 41
 - draw, 42
 - drawSensors, 42
 - getISensorLateral, 42
 - getISensorSemicircle, 42
 - getLateralSide, 43
 - l, 43
 - nSensorsLateral, 44
 - nSensorsSemicircle, 44
 - R, 44
 - setISensorLateral, 43
 - setSensor, 43
 - sinAlpha, 44
- CappedRectangle.cpp, 201
- CappedRectangle.h, 202
- childLinks
 - ArticulatedLink, 29
- Circle, 45
 - ~Circle, 46
 - Circle, 46
 - computeBox, 46
 - computeInertia, 46
 - computeMass, 46
 - detectContacts, 46, 47
 - detectMouseContact, 47
 - draw, 47
 - drawSensors, 48
 - getISensor, 48
 - isCircle, 48
 - R, 48
 - sensorsStartAngle, 49
 - setRadius, 48
 - setSensor, 48
- Circle.cpp, 203
- Circle.h, 204
- circles
 - Spherus, 150

- clear
 - GUI, 89
 - GUIWx, 95
- Color, 50
 - ~Color, 50
 - b, 51
 - Color, 50
 - definitions, 51
 - g, 51
 - r, 51
 - setTransparent, 50
 - transparent, 51
- Color.cpp, 205
- Color.h, 206
- colorBlack
 - GUI, 92
- ColorDefinitions, 52
 - ColorDefinitions, 52
- ColorDefinitions
 - ~ColorDefinitions, 52
 - ColorDefinitions, 52
 - names, 52
 - setColor, 52
- ColorDefinitions.cpp, 207
- ColorDefinitions.h, 208
- ColorDefinitions::ltstr, 53
- ColorDefinitions::ltstr
 - operator(), 53
- ColorDefinitions::SimpleColor, 54
- ColorDefinitions::SimpleColor
 - b, 54
 - g, 54
 - r, 54
 - SimpleColor, 54
- colorTransparent
 - GUI, 92
- colorWhite
 - GUI, 92
- ComposedPhysicalObject, 55
 - ComposedPhysicalObject, 56
- ComposedPhysicalObject
 - ~ComposedPhysicalObject, 56
 - addObject, 56
 - ComposedPhysicalObject, 56
 - computeMassProperties, 57
 - computeMemberPositions, 57
 - deleteObjects, 57
 - detectContacts, 57
 - detectMouseContact, 57
 - draw, 58
 - integrate, 58
 - objects, 60
 - registerPrimitives, 58
 - resetSensors, 58
 - rollback, 59
 - s, 60
 - setFillColor, 59
 - setOutlineColor, 59
 - setSensors, 59
- ComposedPhysicalObject.cpp, 209
- ComposedPhysicalObject.h, 210
- computeBodyDerivatives
 - ArticulatedAgentQuasistatic, 15
- computeBodyDerivativesWithoutContacts
 - ArticulatedAgentQuasistatic, 15
- computeBox
 - CappedRectangle, 40
 - Circle, 46
 - PhysicalObject, 121
- computeCircleProprioception
 - Spherus, 147
- computeContacts
 - ContactSolver, 66
 - Simulator, 135
- computeContactsPreda
 - ContactSolver, 66
- computeDerivatives
 - ArticulatedAgentQuasistatic, 16
 - Elastoid, 76
 - PhysicalObject, 121
 - Spherus, 147
- computeDerivativesWithoutContacts
 - ArticulatedAgentQuasistatic, 16
 - Border, 35
 - Elastoid, 76
 - PhysicalObject, 121
 - Spherus, 147
- computeForceQuasistatic
 - ArticulatedLink, 27
- computeForces
 - ArticulatedAgentQuasistatic, 16
- computeGamma
 - VisualSensor, 181
- computeInertia
 - CappedRectangle, 40
 - Circle, 46
 - PhysicalObject, 121
- computeIStar0
 - ArticulatedComponent, 19
- computeMass
 - CappedRectangle, 40
 - Circle, 46
 - PhysicalObject, 122
- computeMassProperties
 - ComposedPhysicalObject, 57
- computeMemberPositions
 - ComposedPhysicalObject, 57
 - Spherus, 148

- computeMotorTorque
 - ArticulatedLink, 27
- computeSinCos
 - ArticulatedLimb, 23
 - ArticulatedLink, 28
 - Spherus, 148
- computeState
 - Spherus, 148
- computeTotalForces
 - ArticulatedAgentQuasistatic, 16
- contact1
 - GlobalContactInfo, 79
- contact2
 - GlobalContactInfo, 79
- contactDForces
 - ContactSolver, 67
- contactForces
 - ContactSolver, 67
- contactIndexC
 - ContactSolver, 67
- contactIndexTemp
 - ContactSolver, 68
- ContactInfo, 61
 - ContactInfo, 61
- ContactInfo
 - ~ContactInfo, 62
 - alpha, 62
 - ContactInfo, 61
 - force, 62
 - iSensor, 62
 - iSensor2, 62
 - n, 63
 - object, 63
 - p, 63
 - parentObject, 63
 - pxn, 63
 - setupObject, 62
 - sigma, 63
 - type, 64
- ContactInfo.cpp, 211
- ContactInfo.h, 212
 - contactTypeForce, 212
 - contactTypeForceParallel, 212
 - contactTypeTorque, 212
- ContactInfo.h
 - ContactInfoPVector, 212
 - ContactType, 212
- ContactInfoPVector
 - ContactInfo.h, 212
- contactMatrix
 - ContactSolver, 68
- contactMatrixTemp
 - ContactSolver, 68
- contacts
 - ContactSolver, 68
 - PhysicalObject, 127
 - Simulator, 137
- ContactSet
 - ContactSolver.h, 214
- contactSetC
 - ContactSolver.h, 214
- contactSetI
 - ContactSolver.h, 214
- contactSetNC
 - ContactSolver.h, 214
- contactSets
 - ContactSolver, 68
- ContactSolver, 65
 - ContactSolver, 66
- ContactSolver
 - ~ContactSolver, 66
 - computeContacts, 66
 - computeContactsPreda, 66
 - contactDForces, 67
 - contactForces, 67
 - contactIndexC, 67
 - contactIndexTemp, 68
 - contactMatrix, 68
 - contactMatrixTemp, 68
 - contacts, 68
 - contactSets, 68
 - ContactSolver, 66
 - contactVector, 68
 - contactVectorTemp, 68
 - contactVelocities, 69
 - driveToZero, 66
 - init, 66
 - initForcesVelocities, 67
 - maxStep, 67
 - nContacts, 69
 - nContactsMax, 69
 - tempMatrix, 69
 - tempVector1, 69
 - tempVector2, 69
 - uploadForces, 67
 - zeros, 69
- contactSolver
 - Simulator, 138
- ContactSolver.cpp, 213
- ContactSolver.h, 214
 - contactSetC, 214
 - contactSetI, 214
 - contactSetNC, 214
- ContactSolver.h
 - ContactSet, 214
- ContactType
 - ContactInfo.h, 212
- contactTypeForce

- ContactInfo.h, 212
- contactTypeForceParallel
 - ContactInfo.h, 212
- contactTypeTorque
 - ContactInfo.h, 212
- contactVector
 - ContactSolver, 68
- contactVectorTemp
 - ContactSolver, 68
- contactVelocities
 - ContactSolver, 69
- controll
 - Iunctus, 101
 - Pac, 116
 - PhysicalObject, 122
 - Simulator, 135
 - Spherus, 148
- Controller, 71
 - ~Controller, 71
 - advanceTime, 71
 - Controller, 71
 - getOutput, 72
 - input, 72
 - nInputs, 72
 - nOutputs, 72
 - output, 72
 - setInput, 72
- controller
 - Iunctus, 102
 - Pac, 116
 - Spherus, 150
- Controller.cpp, 215
- Controller.h, 216
- coriolis
 - ArticulatedLink, 29
- cos2
 - ArticulatedLimb, 24
 - ArticulatedLink, 29
- cosAlpha
 - CappedRectangle, 43
 - Spherus, 150
- cosSum
 - GUIWx.cpp, 230
- cosTheta
 - ArticulatedLimb, 24
 - ArticulatedLink, 30
- cross
 - Vector2, 171
- data
 - GraphData, 85
 - SpikeGraph, 155
- dataSize
 - GraphData, 85
- SpikeGraph, 155
- dc
 - Graph, 82
 - GUIWx, 97
- DECLARE_CLASS
 - ThyrixMainFrame, 164
- DECLARE_EVENT_TABLE
 - ThyrixMainFrame, 164
- DEFAULT_ETA
 - ArticulatedLink.h, 196
- DEFAULT_K
 - ArticulatedLink.h, 196
- defaultDensity
 - ThyrixParameters, 168
- definitions
 - Color, 51
- degrees
 - MathTools, 108
 - ThyrixParameters.h, 277
- deleteContacts
 - ArticulatedAgentBase, 11
 - ArticulatedLink, 28
 - Elastoid, 76
 - Iunctus, 101
 - PhysicalObject, 122
 - Simulator, 135
 - Spherus, 148
- deleteK
 - ArticulatedComponent, 19
- deleteLimbs
 - ArticulatedAgentBase, 11
- deleteLinks
 - ArticulatedLimb, 23
- deleteObjects
 - ComposedPhysicalObject, 57
 - Simulator, 135
- delta
 - VisualSensor, 182
- deltaThetaMax
 - ArticulatedLink, 30
- detectContacts
 - ArticulatedAgentBase, 11
 - Border, 35
 - CappedRectangle, 41
 - Circle, 46, 47
 - ComposedPhysicalObject, 57
 - Elastoid, 76
 - IunctusSimulator, 103
 - PhysicalObject, 122, 123
 - Simulator, 135
 - Spherus, 149
 - VisualSensor, 181, 182
- detectInternalContacts
 - ArticulatedAgentBase, 11

- Elastoid, 76
- PhysicalObject, 123
- Spherus, 149
- detectMouseContact
 - ArticulatedAgentBase, 11
 - CappedRectangle, 41
 - Circle, 47
 - ComposedPhysicalObject, 57
 - Elastoid, 77
 - PhysicalObject, 123
 - Spherus, 149
- detectTorqueContact
 - ArticulatedLink, 28
- draggedObject
 - Simulator, 138
- draggingPoint
 - Simulator, 138
- draw
 - ArticulatedAgentBase, 12
 - ArticulatedLimb, 23
 - ArticulatedLink, 28
 - Border, 35
 - CappedRectangle, 42
 - Circle, 47
 - ComposedPhysicalObject, 58
 - ElasticLink, 73
 - Elastoid, 77
 - Graph, 82
 - GraphicalObject, 87
 - Iunctus, 101
 - Simulator, 135
 - Spherus, 149
 - SpikeGraph, 154
 - VisualSensor, 182
 - World, 185
- drawArrow
 - GUI, 89
 - GUIWx, 95
- drawBuffer
 - Graph, 82
- drawCappedRectangle
 - GUI, 89
 - GUIWx, 95
- drawCircle
 - GUI, 89
 - GUIWx, 95
- drawContactForces
 - PhysicalObject, 123
- drawForce
 - GUI, 89
 - GUIWx, 95
- drawLine
 - GUI, 90
 - GUIWx, 95
- drawRectangle
 - GUI, 90
 - GUIWx, 96
- drawSensors
 - CappedRectangle, 42
 - Circle, 48
- drawTorque
 - GUI, 90
 - GUIWx, 96
- driveToZero
 - ContactSolver, 66
- dt
 - Integrator, 99
- dtDefault
 - World.h, 285
- DUMMY
 - resource.h, 256
- elasticK
 - Spherus.cpp, 261
- ElasticLink, 73
 - ElasticLink, 73
- ElasticLink
 - ~ElasticLink, 73
 - applyForces, 73
 - draw, 73
 - ElasticLink, 73
 - k, 74
 - length, 74
 - object1, 74
 - object2, 74
- ElasticLink.cpp, 217
- ElasticLink.h, 218
- ElasticLink.h
 - AFX_ELASTICLINK_H_CF42E708_-6848_4CF4_B4DA_19E78FAA2BAC__INCLUDED_, 218
 - ElasticLinkVector, 218
- ElasticLinkVector
 - ElasticLink.h, 218
- Elastoid, 75
 - ~Elastoid, 75
 - addObject, 76
 - computeDerivatives, 76
 - computeDerivativesWithoutContacts, 76
 - deleteContacts, 76
 - detectContacts, 76
 - detectInternalContacts, 76
 - detectMouseContact, 77
 - draw, 77
 - Elastoid, 75
 - integrate, 77
 - links, 78
 - objects, 78

- registerPrimitives, [77](#)
- Elastoid.cpp, [219](#)
- Elastoid.h, [220](#)
 - AFX_ELASTOID_H_DCB64B95_-CC6D_4280_9E1F_242450B50FAF__INCLUDED_, [220](#)
- Entry
 - SimulatorThread, [143](#)
- epsilon
 - VisualSensor, [182](#)
- epsilonContact
 - ThyrixParameters, [168](#)
- eta
 - ArticulatedLink, [30](#)
- EVT_MENU_RANGE
 - ThyrixMainFrame.cpp, [273](#)
- expectedTime
 - SimulatorThread, [144](#)
- extension
 - Spherus, [150](#)
- extensionActivation
 - Spherus, [150](#)
- externalForce
 - PhysicalObject, [127](#)
- externalTorque
 - PhysicalObject, [127](#)
- eye
 - Iunctus, [102](#)
- eyes
 - Spherus, [150](#)
- fillColor
 - GraphicalObject, [87](#)
- fillContactMatrix
 - ArticulatedAgentBase, [12](#)
 - ArticulatedLink, [28](#)
 - Border, [35](#)
 - PhysicalObject, [124](#)
 - Simulator, [136](#)
- force
 - ArticulatedLink, [30](#)
 - ContactInfo, [62](#)
 - GlobalContactInfo, [79](#)
- forceLocal
 - ArticulatedLink, [30](#)
- forwardAccelerations
 - ArticulatedAgentQuasistatic, [16](#)
- forwardKinematics
 - ArticulatedAgentQuasistatic, [17](#)
- frame
 - SimulatorThread, [144](#)
- frameInterval
 - SimulatorThread, [144](#)
- g
 - Color, [51](#)
 - ColorDefinitions::SimpleColor, [54](#)
- gen_state
 - MTRandom, [113](#)
- getDeterminant
 - Matrix3, [110](#)
 - SymmetricMatrix3, [157](#)
- getDouble
 - MTRandom, [113](#)
 - Random, [130](#)
- getDoubleClosed
 - MTRandom, [113](#)
- getDoubleHR
 - MTRandom, [113](#)
- getDoubleOpen
 - MTRandom, [113](#)
- getDt
 - Integrator, [98](#)
 - Simulator, [136](#)
 - World, [185](#)
- getElement
 - Matrix3, [110](#)
- getFloat
 - MTRandom, [113](#)
 - Random, [130](#)
- getInt
 - MTRandom, [113](#)
 - Random, [130](#)
- getISensor
 - Circle, [48](#)
- getISensorLateral
 - CappedRectangle, [42](#)
- getISensorSemicircle
 - CappedRectangle, [42](#)
- getLateralSide
 - CappedRectangle, [43](#)
- getLongInt
 - MTRandom, [113](#)
- getModule
 - Vector2, [171](#)
 - Vector3, [177](#)
- getOutput
 - Controller, [72](#)
- getSquaredModule
 - Vector2, [171](#)
 - Vector3, [177](#)
- getValue
 - GraphData, [84](#)
 - SpikeGraph, [155](#)
- GlobalContactInfo, [79](#)
 - GlobalContactInfo, [79](#)
- GlobalContactInfo
 - alpha, [79](#)

- contact1, 79
- contact2, 79
- force, 79
- GlobalContactInfo, 79
- penetration, 79
- GlobalContactInfo.h, 221
- GlobalContactInfo.h
 - GlobalContactInfoVector, 221
- GlobalContactInfoVector
 - GlobalContactInfo.h, 221
- Graph, 81
 - ~Graph, 81
 - bitmap, 82
 - bufferSize, 82
 - dc, 82
 - draw, 82
 - drawBuffer, 82
 - Graph, 81
 - height, 82
 - max, 82
 - min, 83
 - nSubGraphs, 83
 - push, 82
 - subGraphs, 83
 - width, 83
- Graph.cpp, 222
- Graph.h, 223
 - AFX_GRAPH_H__6AA46D96_39C9_-4C8F_9FCC_3446AE0C8C94__INCLUDED_, 223
- GraphData, 84
 - GraphData, 84
- GraphData
 - ~GraphData, 84
 - bufferSize, 85
 - data, 85
 - dataSize, 85
 - getValue, 84
 - GraphData, 84
 - index, 85
 - push, 84
- GraphData.cpp, 224
- GraphData.h, 225
- GraphData.h
 - AFX_GRAPHDATA_H__035AC771_F046_-4ECA_BCDE_E570A1138AE4__INCLUDED_, 225
- GraphicalObject, 86
 - GraphicalObject, 86
- GraphicalObject
 - ~GraphicalObject, 86
 - draw, 87
 - fillColor, 87
 - GraphicalObject, 86
 - outlineColor, 87
 - setColor, 87
 - setFillColor, 87
 - setOutlineColor, 87
- GraphicalObject.cpp, 226
- GraphicalObject.h, 227
- GUI, 88
 - ~GUI, 89
 - clear, 89
 - colorBlack, 92
 - colorTransparent, 92
 - colorWhite, 92
 - drawArrow, 89
 - drawCappedRectangle, 89
 - drawCircle, 89
 - drawForce, 89
 - drawLine, 90
 - drawRectangle, 90
 - drawTorque, 90
 - GUI, 89
 - inverseMapX, 90
 - inverseMapY, 90
 - mapLen, 90
 - mapX, 91
 - mapY, 91
 - outText, 91
 - outTextStatusBar, 91
 - panX, 92
 - panY, 92
 - setBrushColor, 91
 - setPenColor, 91
 - setZoom, 91
 - signX, 92
 - signY, 92
 - zoom, 92
 - zoomX, 92
 - zoomY, 92
- gui
 - ThyrixMainFrame, 166
- GUI.cpp, 228
- GUI.h, 229
- GUIWx, 94
 - ~GUIWx, 94
 - clear, 95
 - dc, 97
 - drawArrow, 95
 - drawCappedRectangle, 95
 - drawCircle, 95
 - drawForce, 95
 - drawLine, 95
 - drawRectangle, 96
 - drawTorque, 96
 - GUIWx, 94
 - outText, 96

- outTextStatusBar, 96
 - setBrushColor, 96
 - setDC, 96
 - setPenColor, 97
 - statusBar, 97
- GUIWx.cpp, 230
 - cosSum, 230
 - sinSum, 230
- GUIWx.h, 231
- halfEpsilonContact
 - ThyrixParameters, 168
- hasElasticTorque
 - ArticulatedLink, 30
- height
 - Graph, 82
- I
 - PhysicalObject, 127
- index
 - GraphData, 85
 - Pac, 116
 - SpikeGraph, 155
- indexContacts
 - Simulator, 136
- infinity
 - ThyrixParameters, 169
- init
 - ContactSolver, 66
 - IunctusSimulator, 104
- initForcesVelocities
 - ContactSolver, 67
- input
 - Controller, 72
- integrate
 - ArticulatedAgentQuasistatic, 17
 - ArticulatedLink, 28
 - Border, 35
 - ComposedPhysicalObject, 58
 - Elastoid, 77
 - Integrator, 98
 - PhysicalObject, 124
 - Spherus, 149
- Integrator, 98
 - dt, 99
 - getDt, 98
 - integrate, 98
 - Integrator, 98
 - setDt, 98
- integrator
 - Simulator, 138
- Integrator.h, 232
- inverseMapX
 - GUI, 90
- inverseMapY
 - GUI, 90
- isCircle
 - Circle, 48
 - PhysicalObject, 124
- isContact
 - Simulator, 138
- iSensor
 - ContactInfo, 62
- iSensor2
 - ContactInfo, 62
- isLeftMouseButtonPressed
 - World, 186
- isRightMouseButtonPressed
 - World, 186
- IStar
 - ArticulatedComponent, 20
- IStar0
 - ArticulatedComponent, 20
- Iunctus, 100
 - ~Iunctus, 100
 - build, 101
 - controll, 101
 - controller, 102
 - deleteContacts, 101
 - draw, 101
 - eye, 102
 - Iunctus, 100
 - myBody, 102
 - proprioception, 101
- Iunctus.cpp, 233
- Iunctus.h, 234
- IunctusSimulator, 103
 - IunctusSimulator, 103
- IunctusSimulator
 - ~IunctusSimulator, 103
 - agent, 104
 - detectContacts, 103
 - init, 104
 - IunctusSimulator, 103
- IunctusSimulator.cpp, 235
- IunctusSimulator.h, 236
- K
 - ArticulatedComponent, 20
- k
 - ArticulatedLink, 31
 - ElasticLink, 74
- kContact
 - ThyrixParameters, 169
- kMenuCapture
 - ThyrixMainFrame, 163
- kMenuFps30
 - ThyrixMainFrame, 163

- kMenuFps60
 - ThyrixMainFrame, 163
- kMenuFps90
 - ThyrixMainFrame, 163
- kMenuStep
 - ThyrixMainFrame, 163
- kMenuTimeDiv10
 - ThyrixMainFrame, 163
- kMenuTimeMax
 - ThyrixMainFrame, 163
- kMenuTimePause
 - ThyrixMainFrame, 163
- kMenuTimeX1
 - ThyrixMainFrame, 163
- kMenuTimeX10
 - ThyrixMainFrame, 163
- kMenuTimeX100
 - ThyrixMainFrame, 163
- kMenuTimeX3
 - ThyrixMainFrame, 163
- kMenuTimeX30
 - ThyrixMainFrame, 163
- kMouseForce
 - ThyrixParameters, 169
- l
 - ArticulatedLimb, 24
 - ArticulatedLink, 31
 - CappedRectangle, 43
 - Pac, 116
- label
 - ArticulatedLimb, 24
 - PhysicalObject, 127
- length
 - ElasticLink, 74
- limbs
 - ArticulatedAgentBase, 13
- LinkContactInfo, 105
 - LinkContactInfo, 105
- LinkContactInfo
 - ~LinkContactInfo, 105
 - alpha, 105
 - LinkContactInfo, 105
 - operator<, 105
 - thetaFactor, 105
 - v, 106
- LinkContactInfo.cpp, 237
- LinkContactInfo.h, 238
- LinkContactInfo.h
 - LinkContactInfoPVector, 238
 - LinkContactInfoVector, 238
- LinkContactInfoPVector
 - LinkContactInfo.h, 238
- LinkContactInfoVector

- LinkContactInfo.h, 238
- links
 - ArticulatedLimb, 24
 - Elastoid, 78
- luDecompose
 - SystemSolver, 159
- luSubstitute
 - SystemSolver, 159
- m
 - PhysicalObject, 128
- M_PI
 - ThyrixParameters.h, 276
- mapLen
 - GUI, 90
- mapX
 - GUI, 91
- mapY
 - GUI, 91
- MathTools, 107
 - MathTools, 107
- MathTools
 - ~MathTools, 107
 - degrees, 108
 - MathTools, 107
 - max, 107
 - modulus, 107
 - pi, 108
 - setSign, 108
 - sign, 108
 - sqr, 108
- MathTools.cpp, 239
- MathTools.h, 240
- MathTools.h
 - AFX_MATHTOOLS_H_D0539AA7_-
8BC3_4A5F_99A3_730FAE12C22E_-
INCLUDED_, 240
- matrix
 - Matrix3, 111
- Matrix3, 109
 - ~Matrix3, 109
 - getDeterminant, 110
 - getElement, 110
 - matrix, 111
 - Matrix3, 109
 - operator+=", 110
 - operator[], 110
 - setColumn, 110
 - setElement, 110
 - setRow, 110
 - setToZero, 111
- Matrix3.cpp, 241
- Matrix3.h, 242
- max

- Graph, 82
- MathTools, 107
- maxExtension
 - Spherus.cpp, 261
- maxExtensionForce
 - Spherus.cpp, 261
- maxRocketForce
 - Spherus.cpp, 261
- maxStep
 - ContactSolver, 67
- min
 - Graph, 83
- mirror
 - SymmetricMatrix3, 157
- modulus
 - MathTools, 107
- motorTorque
 - ArticulatedLink, 31
- mouseX
 - World, 186
- mouseY
 - World, 186
- MTRandom, 112
 - ~MTRandom, 112
 - gen_state, 113
 - getDouble, 113
 - getDoubleClosed, 113
 - getDoubleHR, 113
 - getDoubleOpen, 113
 - getFloat, 113
 - getInt, 113
 - getLongInt, 113
 - MTRandom, 112
 - operator=, 113
 - p, 114
 - seed, 114
 - state, 114
 - twiddle, 114
- MTRandom.cpp, 243
- MTRandom.h, 244
 - AFX_MTRANDOM_H__80B8318A_-
FECC_423A_9199_FF3E5F3A05D3__-
INCLUDED_, 244
 - MTRandomN, 244
- MTRandomN
 - MTRandom.h, 244
- myBody
 - Iunctus, 102
- n
 - ContactInfo, 63
- names
 - ColorDefinitions, 52
- nCircles
 - Pac, 116
- nContacts
 - ArticulatedComponent, 20
 - ContactSolver, 69
- nContactsMax
 - ContactSolver, 69
- nEffectors
 - Spherus, 150
- nInputs
 - Controller, 72
- nNeurons
 - SpikeGraph, 155
- normal
 - Border, 36
- normalize
 - Vector2, 171
- normalizeAlpha
 - PhysicalObject, 124
- normalizeTheta
 - ArticulatedLink, 29
- nOutputs
 - Controller, 72
- nSensors
 - PhysicalObject, 128
 - Spherus, 151
- nSensorsLateral
 - CappedRectangle, 44
- nSensorsSemicircle
 - CappedRectangle, 44
- nSubGraphs
 - Graph, 83
- nTactileSensors
 - Spherus.cpp, 261
- nVisionSensors
 - Spherus.cpp, 261
- o1
 - Simulator::ObjectPair, 140
- o2
 - Simulator::ObjectPair, 140
- object
 - ContactInfo, 63
- object1
 - ElasticLink, 74
- object2
 - ElasticLink, 74
- ObjectPair
 - Simulator::ObjectPair, 140
- objects
 - ComposedPhysicalObject, 60
 - Elastoid, 78
 - Simulator, 138
- oldInput
 - CachingController, 38

- omega
 - PhysicalObject, 128
- onCapture
 - ThyrixMainFrame, 164
- OnClose
 - ThyrixMainFrame, 164
- OnExit
 - SimulatorThread, 143
- onLeftDown
 - ThyrixMainFrame, 164
- onLeftUp
 - ThyrixMainFrame, 164
- onMotion
 - ThyrixMainFrame, 164
- onMouseLeftDown
 - Simulator, 136
 - World, 185
- onMouseLeftUp
 - Simulator, 136
 - World, 185
- onMouseRightDown
 - World, 185
- onMouseRightUp
 - World, 185
- onRightDown
 - ThyrixMainFrame, 164
- onRightUp
 - ThyrixMainFrame, 164
- onSetFps
 - ThyrixMainFrame, 165
- onSetSpeed
 - ThyrixMainFrame, 165
- onStep
 - ThyrixMainFrame, 165
- operator *
 - Vector2, 171, 172
- operator *=
 - Vector2, 172
 - Vector3, 177
- operator()
 - ColorDefinitions::ltstr, 53
- operator+
 - Vector2, 172
- operator+=
 - Matrix3, 110
 - SymmetricMatrix3, 158
 - Vector2, 172
 - Vector3, 177
- operator-
 - Vector2, 172
- operator-=
 - Vector2, 172
 - Vector3, 177
- operator/=
 - Vector2, 172
 - Vector3, 177
- operator<
 - LinkContactInfo, 105
 - Simulator::ObjectPair, 140
- operator=
 - MTRandom, 113
- operator[]
 - Matrix3, 110
 - Vector2, 172
 - Vector3, 177
- operator^
 - Vector2, 173
- outlineColor
 - GraphicalObject, 87
- output
 - Controller, 72
- outText
 - GUI, 91
 - GUIWx, 96
- outTextStatusBar
 - GUI, 91
 - GUIWx, 96
- p
 - ContactInfo, 63
 - MTRandom, 114
- Pac, 115
 - ~Pac, 115
 - activeLink, 116
 - controll, 116
 - controller, 116
 - index, 116
 - l, 116
 - nCircles, 116
 - Pac, 115
 - rPac, 116
 - u, 117
 - u0, 117
- Pac.cpp, 245
- Pac.h, 246
 - AFX_PAC_H__65EEAA15_5429_4476_-885F_AF955C54055A__INCLUDED_, 246
- PacSimulator, 118
 - PacSimulator, 118
- PacSimulator
 - ~PacSimulator, 118
 - PacSimulator, 118
- PacSimulator.cpp, 247
- PacSimulator.h, 248
- PacSimulator.h

- AFX_PACSIMULATOR_H__E15DFA83_-
B781_468B_8DD3_411BBECD1FF7__-
INCLUDED_, 248
- paint
 - ThyrixMainFrame, 165
- paintDC
 - ThyrixMainFrame, 165
- panX
 - GUI, 92
- panY
 - GUI, 92
- parent
 - PhysicalObject, 128
- parentLink
 - ArticulatedLink, 31
- parentObject
 - ContactInfo, 63
- paused
 - SimulatorThread, 144
- penetration
 - GlobalContactInfo, 79
- PhysicalObject, 119
 - PhysicalObject, 120
- PhysicalObject
 - ~PhysicalObject, 120
 - activations, 126
 - addContact, 121
 - alpha, 126
 - alphaOld, 126
 - boxMax, 126
 - boxMin, 127
 - computeBox, 121
 - computeDerivatives, 121
 - computeDerivativesWithoutContacts, 121
 - computeInertia, 121
 - computeMass, 122
 - contacts, 127
 - controll, 122
 - deleteContacts, 122
 - detectContacts, 122, 123
 - detectInternalContacts, 123
 - detectMouseContact, 123
 - drawContactForces, 123
 - externalForce, 127
 - externalTorque, 127
 - fillContactMatrix, 124
 - I, 127
 - integrate, 124
 - isCircle, 124
 - label, 127
 - m, 128
 - normalizeAlpha, 124
 - nSensors, 128
 - omega, 128
 - parent, 128
 - PhysicalObject, 120
 - r, 128
 - registerPrimitives, 124
 - relativeAlpha, 128
 - relativeR, 129
 - resetSensors, 125
 - rOld, 129
 - rollback, 125
 - saturationForce, 129
 - setMass, 125
 - setPosition, 125
 - setSensor, 125
 - setSensors, 125
 - setVelocity, 126
 - v, 129
- PhysicalObject.cpp, 249
- PhysicalObject.h, 250
- PhysicalObject.h
 - PhysicalObjectPVector, 250
- PhysicalObjectPVector
 - PhysicalObject.h, 250
- pi
 - MathTools, 108
- primitives
 - Simulator, 138
- proprioception
 - Iunctus, 101
- purgeContainer
 - purgeContainer.h, 251
- purgeContainer.h, 251
- purgeContainer.h
 - purgeContainer, 251
- push
 - Graph, 82
 - GraphData, 84
 - SpikeGraph, 155
- pxn
 - ContactInfo, 63
- Q
 - ArticulatedComponent, 20
- R
 - CappedRectangle, 44
 - Circle, 48
 - Spherus, 151
 - VisualSensor, 183
- r
 - Color, 51
 - ColorDefinitions::SimpleColor, 54
 - PhysicalObject, 128
- Random, 130
 - ~Random, 130

- getDouble, 130
 - getFloat, 130
 - getInt, 130
 - Random, 130
 - setSeed, 130
- Random.cpp, 252
- Random.h, 253
- RandomController, 132
 - RandomController, 132
- RandomController
 - ~RandomController, 132
 - advanceTime, 132
 - RandomController, 132
- RandomController.cpp, 254
- RandomController.h, 255
- real
 - ThyrixParameters.h, 277
- realTime
 - Simulator, 138
- registerObject
 - Simulator, 136
- registerPrimitive
 - Simulator, 137
- registerPrimitives
 - ArticulatedAgentBase, 12
 - ComposedPhysicalObject, 58
 - Elastoid, 77
 - PhysicalObject, 124
- relativeAlpha
 - PhysicalObject, 128
- relativeR
 - PhysicalObject, 129
- resetSensors
 - ComposedPhysicalObject, 58
 - PhysicalObject, 125
- resource.h, 256
 - DUMMY, 256
- rocketActivations
 - Spherus, 151
- rocketForces
 - Spherus, 151
- rocketForceValues
 - Spherus, 151
- rOld
 - PhysicalObject, 129
- rollback
 - ArticulatedAgentQuasistatic, 17
 - ArticulatedLink, 29
 - Border, 36
 - ComposedPhysicalObject, 59
 - PhysicalObject, 125
- rotate
 - Vector2, 173
 - Vector3, 177
- rPac
 - Pac, 116
- s
 - ComposedPhysicalObject, 60
- saturationForce
 - PhysicalObject, 129
- seed
 - MTRandom, 114
- sensorsStartAngle
 - Circle, 49
- setBrushColor
 - GUI, 91
 - GUIWx, 96
- setColor
 - ColorDefinitions, 52
 - GraphicalObject, 87
- setColumn
 - Matrix3, 110
- setDC
 - GUIWx, 96
- setDt
 - Integrator, 98
- setElement
 - Matrix3, 110
 - Vector2, 173
 - Vector3, 178
- setElements
 - SymmetricMatrix3, 158
- setFillColor
 - ArticulatedAgentBase, 12
 - ArticulatedLimb, 23
 - ComposedPhysicalObject, 59
 - GraphicalObject, 87
- setFramesPerSecond
 - SimulatorThread, 143
- setInput
 - Controller, 72
- setISensorLateral
 - CappedRectangle, 43
- setMass
 - PhysicalObject, 125
- setMouseCoordinates
 - World, 186
- setMouseForce
 - Simulator, 137
- setOutlineColor
 - ArticulatedAgentBase, 13
 - ArticulatedLimb, 23
 - ComposedPhysicalObject, 59
 - GraphicalObject, 87
- setParentLink
 - ArticulatedLink, 29
- setPause

- SimulatorThread, 143
- setPenColor
 - GUI, 91
 - GUIWx, 97
- setPosition
 - PhysicalObject, 125
- setProperty
 - ArticulatedLimb, 24
- setRadius
 - Circle, 48
- setRow
 - Matrix3, 110
- setSeed
 - Random, 130
- setSensor
 - CappedRectangle, 43
 - Circle, 48
 - PhysicalObject, 125
- setSensors
 - ComposedPhysicalObject, 59
 - PhysicalObject, 125
- setSign
 - MathTools, 108
- setTimeFactor
 - SimulatorThread, 143
- setToZero
 - Matrix3, 111
 - Vector2, 173
 - Vector3, 178
- setTransparent
 - Color, 50
- setup
 - Simulator, 137
- setupObject
 - ContactInfo, 62
- setVelocity
 - PhysicalObject, 126
- setXY
 - Vector2, 173
- setXYZ
 - Vector3, 178
- setZoom
 - GUI, 91
- sigma
 - ContactInfo, 63
- sign
 - MathTools, 108
- signX
 - GUI, 92
- signY
 - GUI, 92
- SimpleColor
 - ColorDefinitions::SimpleColor, 54
- simulationTimer
 - SimulatorThread, 144
- Simulator, 133
 - ~Simulator, 134
 - advanceTime, 134
 - applyMouseForce, 134
 - computeContacts, 135
 - contacts, 137
 - contactSolver, 138
 - controll, 135
 - deleteContacts, 135
 - deleteObjects, 135
 - detectContacts, 135
 - draggedObject, 138
 - draggingPoint, 138
 - draw, 135
 - fillContactMatrix, 136
 - getDt, 136
 - indexContacts, 136
 - integrator, 138
 - isContact, 138
 - objects, 138
 - onMouseLeftDown, 136
 - onMouseLeftUp, 136
 - primitives, 138
 - realTime, 138
 - registerObject, 136
 - registerPrimitive, 137
 - setMouseForce, 137
 - setup, 137
 - Simulator, 134
 - sortedObjects, 139
 - sortObjects, 137
 - unsetMouseForce, 137
 - xContacts, 139
 - xyContacts, 139
- Simulator.cpp, 257
- Simulator.h, 258
- Simulator::ObjectPair, 140
- Simulator::ObjectPair
 - o1, 140
 - o2, 140
 - ObjectPair, 140
 - operator<, 140
- SimulatorThread, 142
 - SimulatorThread, 142
- SimulatorThread
 - ~SimulatorThread, 142
 - Entry, 143
 - expectedTime, 144
 - frame, 144
 - frameInterval, 144
 - OnExit, 143
 - paused, 144
 - setFramesPerSecond, 143

- setPause, 143
 - setTimeFactor, 143
 - simulationTimer, 144
 - SimulatorThread, 142
 - sleepIfAhead, 143
 - step, 143
 - timeFactor, 144
 - world, 144
- SimulatorThread.cpp, 259
- SimulatorThread.h, 260
- sin2
 - ArticulatedLimb, 25
 - ArticulatedLink, 31
- sinAlpha
 - CappedRectangle, 44
 - Spherus, 151
- sinCos
 - ArticulatedLimb, 25
 - ArticulatedLink, 31
- sinSum
 - GUIWx.cpp, 230
- sinTheta
 - ArticulatedLimb, 25
 - ArticulatedLink, 31
- sleepIfAhead
 - SimulatorThread, 143
- solveSystem
 - ArticulatedAgentBase, 13
- sortedObjects
 - Simulator, 139
- sortObjects
 - Simulator, 137
- Spherus, 146
 - ~Spherus, 147
 - circles, 150
 - computeCircleProprioception, 147
 - computeDerivatives, 147
 - computeDerivativesWithoutContacts, 147
 - computeMemberPositions, 148
 - computeSinCos, 148
 - computeState, 148
 - controll, 148
 - controller, 150
 - cosAlpha, 150
 - deleteContacts, 148
 - detectContacts, 149
 - detectInternalContacts, 149
 - detectMouseContact, 149
 - draw, 149
 - extension, 150
 - extensionActivation, 150
 - eyes, 150
 - integrate, 149
 - nEffectors, 150
 - nSensors, 151
 - R, 151
 - rocketActivations, 151
 - rocketForces, 151
 - rocketForceValues, 151
 - sinAlpha, 151
 - Spherus, 147
 - targetExtension, 151
- Spherus.cpp, 261
 - elasticK, 261
 - maxExtension, 261
 - maxExtensionForce, 261
 - maxRocketForce, 261
 - nTactileSensors, 261
 - nVisionSensors, 261
- Spherus.h, 262
- SpherusSimulator, 153
 - SpherusSimulator, 153
- SpherusSimulator
 - ~SpherusSimulator, 153
 - agent, 153
 - SpherusSimulator, 153
- SpherusSimulator.cpp, 263
- SpherusSimulator.h, 264
- SpikeGraph, 154
 - SpikeGraph, 154
- SpikeGraph
 - ~SpikeGraph, 154
 - advanceTime, 154
 - bufferSize, 155
 - data, 155
 - dataSize, 155
 - draw, 154
 - getValue, 155
 - index, 155
 - nNeurons, 155
 - push, 155
 - SpikeGraph, 154
- SpikeGraph.cpp, 265
- SpikeGraph.h, 266
- SpikeGraph.h
 - AFX_SPIKEGRAPH_H_BB61CB4C_-
B0D5_4444_BFA1_4088183CC669__-
INCLUDED_, 266
- sqr
 - MathTools, 108
 - ThyrixParameters.h, 277
- start
 - ThyrixMainFrame, 165
- state
 - MTRandom, 114
- statusBar
 - GUIWx, 97
- step

- SimulatorThread, 143
- subGraphs
 - Graph, 83
- svDecompose
 - SystemSolver, 160
- svSubstitute
 - SystemSolver, 160
- SymmetricMatrix3, 157
 - SymmetricMatrix3, 157
- SymmetricMatrix3
 - ~SymmetricMatrix3, 157
 - getDeterminant, 157
 - mirror, 157
 - operator+=", 158
 - setElements, 158
 - SymmetricMatrix3, 157
- SymmetricMatrix3.cpp, 267
- SymmetricMatrix3.h, 268
- SystemSolver, 159
 - SystemSolver, 159
- SystemSolver
 - ~SystemSolver, 159
 - luDecompose, 159
 - luSubstitute, 159
 - svDecompose, 160
 - svSubstitute, 160
 - SystemSolver, 159
- SystemSolver.cpp, 269
- SystemSolver.cpp
 - TINY, 269
- SystemSolver.h, 270
- targetExtension
 - Spherus, 151
- tempMatrix
 - ContactSolver, 69
- tempVector1
 - ContactSolver, 69
- tempVector2
 - ContactSolver, 69
- theta
 - ArticulatedLimb, 25
 - ArticulatedLink, 32
- theta0
 - ArticulatedLink, 32
- theta0Max
 - ArticulatedLink, 32
- theta0Min
 - ArticulatedLink, 32
- thetaD
 - ArticulatedLink, 32
- thetaDD
 - ArticulatedLink, 32
- thetaFactor
 - LinkContactInfo, 105
- thetaOld
 - ArticulatedLink, 33
- thread
 - ThyrixMainFrame, 166
- ThyrixApplication, 161
 - ThyrixApplication, 161
- ThyrixApplication
 - ~ThyrixApplication, 161
 - ThyrixApplication, 161
 - world, 161
- ThyrixApplication.cpp, 271
- ThyrixApplication.h, 272
- ThyrixMainFrame, 162
 - kMenuCapture, 163
 - kMenuFps30, 163
 - kMenuFps60, 163
 - kMenuFps90, 163
 - kMenuStep, 163
 - kMenuTimeDiv10, 163
 - kMenuTimeMax, 163
 - kMenuTimePause, 163
 - kMenuTimeX1, 163
 - kMenuTimeX10, 163
 - kMenuTimeX100, 163
 - kMenuTimeX3, 163
 - kMenuTimeX30, 163
 - ThyrixMainFrame, 163
 - Unused, 163
- ThyrixMainFrame
 - ~ThyrixMainFrame, 163
 - bitmap, 166
 - buffer, 166
 - bufferHeight, 166
 - bufferWidth, 166
 - buildMenu, 164
 - DECLARE_CLASS, 164
 - DECLARE_EVENT_TABLE, 164
 - gui, 166
 - onCapture, 164
 - OnClose, 164
 - onLeftDown, 164
 - onLeftUp, 164
 - onMotion, 164
 - onRightDown, 164
 - onRightUp, 164
 - onSetFps, 165
 - onSetSpeed, 165
 - onStep, 165
 - paint, 165
 - paintDC, 165
 - start, 165
 - thread, 166
 - ThyrixMainFrame, 163

- world, 166
- zoom, 166
- ThyrixMainFrame.cpp, 273
- ThyrixMainFrame.cpp
 - EVT_MENU_RANGE, 273
- ThyrixMainFrame.h, 274
- ThyrixParameters, 168
 - ThyrixParameters, 168
- ThyrixParameters
 - ~ThyrixParameters, 168
 - defaultDensity, 168
 - epsilonContact, 168
 - halfEpsilonContact, 168
 - infinity, 169
 - kContact, 169
 - kMouseForce, 169
 - ThyrixParameters, 168
- ThyrixParameters.cpp, 275
- ThyrixParameters.h, 276
- ThyrixParameters.h
 - AFX_THYRIXPARAMETERS_H_-
_442475F3_9963_4247_8192_-
1505B7F7E4D5__INCLUDED_, 276
 - degrees, 277
 - M_PI, 276
 - real, 277
 - sqr, 277
- timeFactor
 - SimulatorThread, 144
- timeStep
 - World, 186
- TINY
 - SystemSolver.cpp, 269
- torqueContact
 - ArticulatedLink, 33
- totalForce
 - ArticulatedLink, 33
- totalTorque
 - ArticulatedLink, 33
- transparent
 - Color, 51
- twiddle
 - MTRandom, 114
- type
 - ContactInfo, 64
- u
 - Pac, 117
- u0
 - Pac, 117
- unsetMouseForce
 - Simulator, 137
- Unused
 - ThyrixMainFrame, 163
- updateMax
 - Vector2, 174
- updateMin
 - Vector2, 174
- uploadForces
 - ContactSolver, 67
- v
 - LinkContactInfo, 106
 - PhysicalObject, 129
- Vector2, 170
 - ~Vector2, 171
 - cross, 171
 - getModule, 171
 - getSquaredModule, 171
 - normalize, 171
 - operator *, 171, 172
 - operator ==, 172
 - operator +, 172
 - operator +=, 172
 - operator -, 172
 - operator =, 172
 - operator /=, 172
 - operator [], 172
 - operator ^, 173
 - rotate, 173
 - setElement, 173
 - setToZero, 173
 - setXY, 173
 - updateMax, 174
 - updateMin, 174
 - Vector2, 170
 - x, 174
 - y, 174
- Vector2.cpp, 278
- Vector2.h, 279
- Vector3, 176
 - ~Vector3, 176
 - getModule, 177
 - getSquaredModule, 177
 - operator ==, 177
 - operator +=, 177
 - operator =, 177
 - operator /=, 177
 - operator [], 177
 - rotate, 177
 - setElement, 178
 - setToZero, 178
 - setXYZ, 178
 - Vector3, 176
 - x, 178
 - y, 178
 - z, 178
- Vector3.cpp, 280

- Vector3.h, [281](#)
- viewAngle
 - VisualSensor, [183](#)
- VisualSensor, [180](#)
 - VisualSensor, [181](#)
- VisualSensor
 - ~VisualSensor, [181](#)
 - activate, [181](#)
 - computeGamma, [181](#)
 - delta, [182](#)
 - detectContacts, [181](#), [182](#)
 - draw, [182](#)
 - epsilon, [182](#)
 - R, [183](#)
 - viewAngle, [183](#)
 - VisualSensor, [181](#)
- VisualSensor.cpp, [282](#)
- VisualSensor.h, [283](#)
- vLocal
 - ArticulatedComponent, [20](#)
- width
 - Graph, [83](#)
- World, [184](#)
 - ~World, [184](#)
 - advanceTime, [185](#)
 - draw, [185](#)
 - getDt, [185](#)
 - isLeftMouseButtonPressed, [186](#)
 - isRightMouseButtonPressed, [186](#)
 - mouseX, [186](#)
 - mouseY, [186](#)
 - onMouseLeftDown, [185](#)
 - onMouseLeftUp, [185](#)
 - onMouseRightDown, [185](#)
 - onMouseRightUp, [185](#)
 - setMouseCoordinates, [186](#)
 - timeStep, [186](#)
 - World, [184](#)
- world
 - SimulatorThread, [144](#)
 - ThyrixApplication, [161](#)
 - ThyrixMainFrame, [166](#)
- World.cpp, [284](#)
- World.h, [285](#)
 - AFX_WORLD_H_A1A33537_BF98_-
497E_B4CF_3A2839E9C233__-
INCLUDED_, [285](#)
 - dtDefault, [285](#)
- wxIncludes.h, [286](#)
- x
 - Vector2, [174](#)
 - Vector3, [178](#)
- xContacts
 - Simulator, [139](#)
- xyContacts
 - Simulator, [139](#)
- y
 - Vector2, [174](#)
 - Vector3, [178](#)
- z
 - Vector3, [178](#)
- zeros
 - ContactSolver, [69](#)
- zoom
 - GUI, [92](#)
 - ThyrixMainFrame, [166](#)
- zoomX
 - GUI, [92](#)
- zoomY
 - GUI, [92](#)