RobotSwarmSimulator

Generated by Doxygen 1.5.6

Tue Feb 3 11:11:10 2009

Contents

Chapter 1

Code Documentation of the Robot Swarm Simulator

1.1 Introduction

This Simulator is developed within the project group Schlaue Schwaerme.

Chapter 2

Directory Hierarchy

2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 4

Class Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

Directory Documentation

6.1 src/ActivationSequenceGenerators/ Directory Reference

Files

- file **activation_sequence_generator.h**
- file **asynchronous_asg.cc**
- file **asynchronous_asg.h**
- file **synchronous_asg.cc**
- file **synchronous_asg.h**

6.2 src/EventHandlers/ Directory Reference

Files

- file **event_handler.cc**
- file **event_handler.h**
- file **marker_request_handler.cc**
- file **marker_request_handler.h**
- file **request_handler.cc**
- file **request_handler.h**
- file **type_change_request_handler.cc**
- file **type_change_request_handler.h**
- file **vector_request_handler.cc**
- file **vector_request_handler.h**

6.3 src/Events/ Directory Reference

Files

- file **compute_event.cc**
- file **compute_event.h**
- file **event.cc**
- file **event.h**
- file **handle_requests_event.cc**
- file **handle_requests_event.h**
- file **look_event.cc**
- file **look_event.h**

6.4 src/Model/ Directory Reference

Files

- file **box.cc**
- file **box.h**
- file **box_identifier.cc**
- file **box_identifier.h**
- file **identifier.cc**
- file **identifier.h**
- file **marker_identifier.cc**
- file **marker_identifier.h**
- file **marker_information.cc**
- file **marker_information.h**
- file **obstacle.cc**
- file **obstacle.h**
- file **obstacle_identifier.cc**
- file **obstacle_identifier.h**
- file **robot.cc**
- file **robot.h**
- file **robot_data.cc**
- file **robot_data.h**
- file **robot_identifier.cc**
- file **robot_identifier.h**
- file **sphere.cc**
- file **sphere.h**
- file **sphere_identifier.cc**
- file **sphere_identifier.h**
- file **world_information.cc**
- file **world_information.h**
- file **world_object.cc**
- file **world_object.h**

6.5 src/OpenGL/ Directory Reference

Files

- file **GLHeaders.h**
- file **GLUHeaders.h**
- file **GLUTHeaders.h**
- file **PgGLUT.h**

6.6 src/Requests/ Directory Reference

Files

- file **acceleration_request.h**
- file **marker_request.h**
- file **position_request.h**
- file **request.cc**
- file **request.h**
- file **type_change_request.h**
- file **vector_request.h**
- file **velocity_request.h**

6.7 src/SimulationControl/ Directory Reference

Files

- file **glut_gui.cc**
- file **glut_gui.h**
- file **gui.cc**
- file **gui.h**
- file **history.cc**
- file **history.h**
- file **simulation_control.cc**
- file **simulation_control.h**
- file **visualizer.h**

6.8 src/SimulationKernel/ Directory Reference

Files

- file **robot_control.cc**
- file **robot_control.h**
- file **simulation_kernel.cc**
- file **simulation_kernel.h**
- file **simulation_listener.h**

6.9 src/ Directory Reference

Directories

- directory ActivationSequenceGenerators
- directory EventHandlers
- directory Events
- directory Model
- directory OpenGL
- directory Requests
- directory SimulationControl
- directory SimulationKernel
- directory Statistics
- directory UserInterfaces
- directory Utilities
- directory Views
- directory Visualisation

Files

- file **testheader.cc**
- file **testheader.h**
- file **testheaderB.cc**
- file **testheaderB.h**

6.10 src/Statistics/ Directory Reference

Files

- file **numset_stats.cc**
- file **numset_stats.h**
- file **vecset_stats.cc**
- file **vecset_stats.h**

6.11 src/UserInterfaces/ Directory Reference

Files

- file **SampleGUI.cc**
- file **SampleGUI.h**

6.12 src/Utilities/ Directory Reference

Files

- file **coord_converter.cc**
- file **coord_converter.h**
- file **distribution_generator.cc**
- file **distribution_generator.h**
- file **parser.cc**
- file **parser.h**
- file **szenario_generator.cc**
- file **szenario_generator.h**
- file **unsupported_operation_exception.h**
- file **vector3d.h**

6.13 src/Views/ Directory Reference

Files

- file **abstract_view_factory.h**
- file **acceleration_view.cc**
- file **acceleration_view.h**
- file **box_view.cc**
- file **box_view.h**
- file **chain_view.cc**
- file **chain_view.h**
- file **cog_view.cc**
- file **cog_view.h**
- file **coordinate_system_view.cc**
- file **coordinate_system_view.h**
- file **full_view.cc**
- file **full_view.h**
- file **global_view.cc**
- file **global_view.h**
- file **id_view.cc**
- file **id_view.h**
- file **marker_information_view.cc**
- file **marker_information_view.h**
- file **neighbor_view.cc**
- file **neighbor_view.h**
- file **octree.cc**
- file **octree.h**
- file **octree_utilities.cc**
- file **octree_utilities.h**
- file **octree_view.cc**
- file **octree_view.h**
- file **one_point_formation_view.cc**
- file **one_point_formation_view.h**
- file **own_acceleration_view.cc**
- file **own_acceleration_view.h**
- file **own_coordinate_system_view.cc**
- file **own_coordinate_system_view.h**
- file **own_id_view.cc**
- file **own_id_view.h**
- file **own_marker_view.cc**
- file **own_marker_view.h**
- file **own_position_view.cc**
- file **own_position_view.h**
- file **own_status_view.cc**
- file **own_status_view.h**
- file **own_type_view.cc**

- file **own_type_view.h**
- file **own_velocity_view.cc**
- file **own_velocity_view.h**
- file **parametrized_view_factory.h**
- file **point_in_obstacle_view.cc**
- file **point_in_obstacle_view.h**
- file **position_view.cc**
- file **position_view.h**
- file **robot_status_view.cc**
- file **robot_status_view.h**
- file **robot_type_view.cc**
- file **robot_type_view.h**
- file **self_view.cc**
- file **self_view.h**
- file **sphere_view.cc**
- file **sphere_view.h**
- file **spheric_view.cc**
- file **spheric_view.h**
- file **velocity_view.cc**
- file **velocity_view.h**
- file **view.cc**
- file **view.h**
- file **view_factory.h**

6.14 src/Visualisation/ Directory Reference

Files

- file **camera.cc**
- file **camera.h**
- file **follow_swarm_camera.cc**
- file **follow_swarm_camera.h**
- file **font.h**
- file **moveable_camera.cc**
- file **moveable_camera.h**
- file **simulation_renderer.cc**
- file **simulation_renderer.h**

Chapter 7

Namespace Documentation

7.1 CoordConverter Namespace Reference

Namespace providing utility methods to convert global to local coordinates and vice versa.

Functions

- boost::shared_ptr< Vector3d > global_to_local (const Vector3d &absolute_coord, const Vector3d &origin, const boost::tuple< boost::shared_ptr< Vector3d >, boost::shared_ptr< Vector3d >, boost::shared_ptr< Vector3d > > &local_coord_system)
- boost::shared_ptr< Vector3d > local_to_global (const Vector3d &local_coord, const Vector3d &local_origin, const boost::tuple< boost::shared_ptr< const Vector3d >, boost::shared_ptr< const Vector3d >, boost::shared_ptr< const Vector3d > > &local_coord_system)

7.1.1 Detailed Description

Namespace providing utility methods to convert global to local coordinates and vice versa.

7.1.2 Function Documentation

7.1.2.1 boost::shared_ptr< Vector3d > CoordConverter::global_to_local (const Vector3d & *absolute_coord*, const Vector3d & *origin*, const boost::tuple< boost::shared_ptr< Vector3d >, boost::shared_ptr< Vector3d >, boost::shared_ptr< Vector3d > > & *local_coord_system*)

calculates local coordinates from global ones

Parameters:

absolute_coord,: the global cordinate

origin,: the origin in global coordinates (typically a robot position)

local_coord_system,: three base vectors

Returns:

the local coordinate

Definition at line 11 of file coord_converter.cc.

Referenced by View::get_position().

7.1.2.2 boost::shared_ptr< Vector3d > CoordConverter::local_to_global (const Vector3d & *local_coord*, const Vector3d & *local_origin*, const boost::tuple< boost::shared_ptr< const Vector3d >, boost::shared_ptr< const Vector3d >, boost::shared_ptr< const Vector3d > > & *local_coord_system*)

calculates global coordinates from local ones. The origin of global systems is assumed to be (0,0,0). The global coordinate system is assumed to use unit vectors as base.

Parameters:

local_coord,: the local coordinate

local_coord_system,: the three base vectors of the local coordinate system

Returns:

the global coordinate

Definition at line 30 of file coord_converter.cc.

7.2 PgGLUT Namespace Reference

C++ wrapper for the OpenGL Toolkit (GLUT) Interface.

Functions

- void **display_callback_delegate** ()
- void **keyboard_callback_delegate** (unsigned char key, int x, int y)
- void **mouse_callback_delegate** (int button, int state, int x, int y)
- void **reshape_callback_delegate** (int width, int height)
- void init (const std::string &window_name, int &argc, char **argv, int width=500, int height=500, int x=0, int y=0)

  Initializes the GLUT OpenGL context.

- void glutMainLoop ()

  Starts the GLUT main loop.

- void glutDisplayFunc (boost::function< void()> func)

  Wrapper around GLUT method (supports class member methods).

- void glutKeyboardFunc (boost::function< void(unsigned char key, int x, int y)> func)

  Wrapper around GLUT method (supports class member methods).

- void glutMouseFunc (boost::function< void(int button, int state, int x, int y)> func)

  Wrapper around GLUT method (supports class member methods).

- void glutReshapeFunc (boost::function< void(int width, int height)> func)

  Wrapper around GLUT method (supports class member methods).

Variables

- bool **initialized** = false
- boost::function< void()> **display_callback_**
- boost::function< void(int width, int height)> **reshape_callback_**
- boost::function< void(unsigned char key, int x, int y)> **keyboard_callback_**
- boost::function< void(int button, int state, int x, int y)> **mouse_callback_**

7.2.1 Detailed Description

C++ wrapper for the OpenGL Toolkit (GLUT) Interface.

For the documentation of the 'glut*Func(...)' callbacks, see the GLUT documentation. Note that this wrapper supports class member methods as callbacks.

Currently supported callbacks:

- display callback
- keyboard callback
- mouse callback
- reshape callback

See also:

http://www.opengl.org/resources/libraries/glut/spec3/spec3.html

7.2.2 Function Documentation

7.2.2.1 void PgGLUT::glutDisplayFunc (boost::function< void()> *func*)

Wrapper around GLUT method (supports class member methods).

See also:

http://www.opengl.org/resources/libraries/glut/spec3/spec3.html

Definition at line 88 of file PgGLUT.h.

7.2.2.2 void PgGLUT::glutKeyboardFunc (boost::function< void(unsigned char key, int x, int y)> *func*)

Wrapper around GLUT method (supports class member methods).

See also:

http://www.opengl.org/resources/libraries/glut/spec3/spec3.html

Definition at line 97 of file PgGLUT.h.

7.2.2.3 void PgGLUT::glutMainLoop ()

Starts the GLUT main loop.

Checks wether GLUT was initialized using PgGLUT::init. If so, the main loop is entered. If not, a warning is issued and the method returns.

Definition at line 76 of file PgGLUT.h.

7.2.2.4 void PgGLUT::glutMouseFunc (boost::function< void(int button, int state, int x, int y)> *func*)

Wrapper around GLUT method (supports class member methods).

See also:

http://www.opengl.org/resources/libraries/glut/spec3/spec3.html

Definition at line 106 of file PgGLUT.h.

7.2.2.5 void PgGLUT::glutReshapeFunc (boost::function< void(int width, int height)> *func*)

Wrapper around GLUT method (supports class member methods).

See also:

http://www.opengl.org/resources/libraries/glut/spec3/spec3.html

Definition at line 115 of file PgGLUT.h.

7.2.2.6 void PgGLUT::init (const std::string & *window_name*, int & *argc*, char ** *argv*, int *width* = 500, int *height* = 500, int *x* = 0, int *y* = 0)

Initializes the GLUT OpenGL context.

Note that calling this method more than once has no effect.

See also:

http://www.opengl.org/resources/libraries/glut/spec3/spec3.html

Definition at line 60 of file PgGLUT.h.

Chapter 8

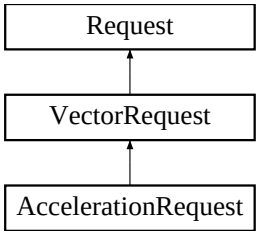
Class Documentation

8.1 AccelerationRequest Class Reference

An Acceleration Request is issued by a robot which wants to change its acceleration to a new value.

`#include <acceleration_request.h>`

Inheritance diagram for AccelerationRequest::



Public Member Functions

- **AccelerationRequest** (Robot &robot, boost::shared_ptr< Vector3d > requested_vector)

8.1.1 Detailed Description

An Acceleration Request is issued by a robot which wants to change its acceleration to a new value.

Notes: The new acceleration is expressed in terms of the local coordinate system of the robot. This means it has to be transformed before using.

The request cannot be changed after construction.

Definition at line 23 of file acceleration_request.h.

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

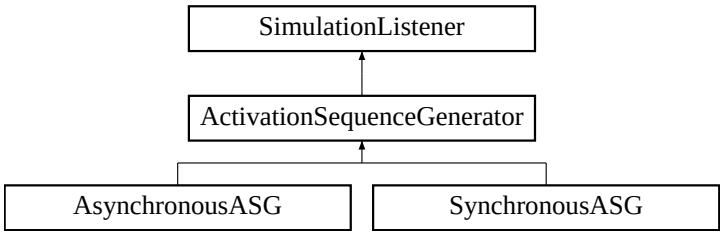
- src/Requests/acceleration_request.h

8.2 ActivationSequenceGenerator Class Reference

Interface for activation sequence generators.

`#include <activation_sequence_generator.h>`

Inheritance diagram for ActivationSequenceGenerator::



Public Member Functions

- virtual void initialize (const History &history, const vector< boost::shared_ptr< Robot > > &robots)=0
- virtual boost::shared_ptr< Event > get_next_event ()=0
- virtual int get_time_of_next_event ()=0

8.2.1 Detailed Description

Interface for activation sequence generators.

The activation sequence generator (AGS) decides, according to some user–specified rules, how to time the different events. In other words, it manages the timing of the execution of the robot algorithms.

The ActivationSequenceGenerator class inherits from the SimulationListener interface.

Definition at line 33 of file activation_sequence_generator.h.

8.2.2 Member Function Documentation

8.2.2.1 virtual void ActivationSequenceGenerator::initialize (const History & *history*, const vector< boost::shared_ptr< Robot > > & *robots*) [pure virtual]

Initializes the ASG.

Parameters:

The history

Implemented in AsynchronousASG, and SynchronousASG.

8.2.2.2 virtual boost::shared_ptr<Event> ActivationSequenceGenerator::get_next_event () [pure virtual]

Returns the next event.

Returns:

The next event produced by the ASG

Implemented in AsynchronousASG, and SynchronousASG.

8.2.2.3 virtual int ActivationSequenceGenerator::get_time_of_next_event () [pure virtual]

Returns the time the next event happens

Returns:

Integer representing the next time an event will happen

Implemented in AsynchronousASG, and SynchronousASG.

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

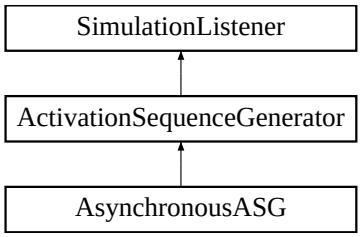
- src/ActivationSequenceGenerators/activation_sequence_generator.h

8.3 AsynchronousASG Class Reference

The asynchronous ASG tries to produce a sequence of events challenging to algorithms developed for the asynchronous time model. Nevertheless it is of course not equivalent to the asynchronous time model.

`#include <asynchronous_asg.h>`

Inheritance diagram for AsynchronousASG::



Public Member Functions

- **AsynchronousASG** (unsigned int seed, double participation_probability, double lambda)
- void initialize (const History &history, const vector< boost::shared_ptr< Robot > > &robots)
- boost::shared_ptr< Event > get_next_event ()
- int get_time_of_next_event ()
- void update (const WorldInformation &world_information, boost::shared_ptr< Event > event)

Private Member Functions

- boost::shared_ptr< Event > choose_event_type ()

Private Attributes

- int time_of_next_event_
- list< boost::shared_ptr< const Request > > unhandled_request_set_
- list< boost::shared_ptr< Robot > > looking_robots_
- list< boost::shared_ptr< Robot > > computing_robots_
- list< boost::shared_ptr< Robot > > handling_robots_
- boost::shared_ptr< DistributionGenerator > distribution_generator_

Friends

- class AsynchronousASGTestAccessor

8.3.1 Detailed Description

The asynchronous ASG tries to produce a sequence of events challenging to algorithms developed for the asynchronous time model. Nevertheless it is of course not equivalent to the asynchronous time model.

The sequence produced by the asynchronous ASG satisfies the following invariants: 1. There are never two events for the same point in time. 2. The order of events for a fixed robot will always be: Look-Compute-HandleRequests 3. In a infinite sequence each robot looks, computes and moves an infinite number of times.

Definition at line 38 of file asynchronous_asg.h.

8.3.2 Member Function Documentation

8.3.2.1 void AsynchronousASG::initialize (const History & *history*, const vector< boost::shared_ptr< Robot > > & *robots*) [virtual]

initializes the asynchronous ASG from the given intial world_state. Needs to be called before the ASG is used

Parameters:

The intial world state

Implements ActivationSequenceGenerator.

Definition at line 41 of file asynchronous_asg.cc.

References looking_robots_.

8.3.2.2 boost::shared_ptr< Event > AsynchronousASG::get_next_event () [virtual]

Returns the next event.

Returns:

The next event in the sequence.

Implements ActivationSequenceGenerator.

Definition at line 49 of file asynchronous_asg.cc.

References choose_event_type(), computing_robots_, distribution_generator_, handling_robots_, looking_robots_, time_of_next_event_, and unhandled_request_set_.

8.3.2.3 int AsynchronousASG::get_time_of_next_event () [inline, virtual]

Returns the time of the next event. This is computed according to...

Returns:

The time of the next event.

Implements ActivationSequenceGenerator.

Definition at line 66 of file asynchronous_asg.h.

References time_of_next_event_.

8.3.2.4 void AsynchronousASG::update (const WorldInformation & *world_information*, boost::shared_ptr< Event > *event*) [virtual]

Updates the sequence of events. Ensures that the events for each fixed robot are in the right order

Parameters:

A constant refrence to the newest world information

The last handled event

Implements SimulationListener.

Definition at line 115 of file asynchronous_asg.cc.

References unhandled_request_set_.

8.3.2.5 boost::shared_ptr< Event > AsynchronousASG::choose_event_type () [private]

chooses, which kind of event will happen next.

Definition at line 125 of file asynchronous_asg.cc.

References computing_robots_, distribution_generator_, handling_robots_, looking_robots_, and time_of_next_event_.

Referenced by get_next_event().

8.3.3 Friends And Related Function Documentation

8.3.3.1 friend class AsynchronousASGTestAccessor [friend]

declare a friend class for doing unit tests with this class.

Definition at line 43 of file asynchronous_asg.h.

8.3.4 Member Data Documentation

8.3.4.1 int AsynchronousASG::time_of_next_event_ [private]

The time the next event will happen.

Definition at line 86 of file asynchronous_asg.h.

Referenced by choose_event_type(), get_next_event(), and get_time_of_next_event().

8.3.4.2 list<boost::shared_ptr<const Request> > AsynchronousASG::unhandled_request_set_ [private]

A set of unhandled requests from some compute events in the past.

Definition at line 91 of file asynchronous_asg.h.

Referenced by get_next_event(), and update().

8.3.4.3 list<boost::shared_ptr<Robot> > AsynchronousASG::looking_robots_ [private]

The set of all robots which will have a look event as their next event.

Definition at line 96 of file asynchronous_asg.h.

Referenced by choose_event_type(), get_next_event(), and initialize().

8.3.4.4 list<boost::shared_ptr<Robot> > AsynchronousASG::computing_robots_ [private]

The set of all robots which will have a compute event as their next event.

Definition at line 101 of file asynchronous_asg.h.

Referenced by choose_event_type(), and get_next_event().

8.3.4.5 list<boost::shared_ptr<Robot> > AsynchronousASG::handling_robots_ [private]

The set of all robots which will have handle requests event as their next event.

Definition at line 106 of file asynchronous_asg.h.

Referenced by choose_event_type(), and get_next_event().

8.3.4.6 boost::shared_ptr<DistributionGenerator> AsynchronousASG::distribution_generator_ [private]

a source of randomness

Definition at line 111 of file asynchronous_asg.h.

Referenced by choose_event_type(), and get_next_event().

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

- src/ActivationSequenceGenerators/asynchronous_asg.h

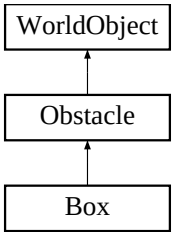
- src/ActivationSequenceGenerators/asynchronous_asg.cc

8.4 Box Class Reference

Denotes an box-obstacle.

`#include <box.h>`

Inheritance diagram for Box::



Public Member Functions

- **Box** (boost::shared_ptr< Identifier > id, boost::shared_ptr< Vector3d > position, double depth, double width, double height)
- **Box** (boost::shared_ptr< Identifier > id, boost::shared_ptr< Vector3d > position, boost::shared_ptr< MarkerInformation > marker_information, double depth, double width, double height)
- double height () const
- void set_height (double new_height)
- double depth () const
- void set_depth (double new_depth)
- double width () const
- void set_width (double new_width)
- bool contains_point (boost::shared_ptr< Vector3d > point) const
- virtual boost::shared_ptr< WorldObject > clone () const

Private Attributes

- double **depth_**
- double **width_**
- double **height_**

8.4.1 Detailed Description

Denotes an box-obstacle.

Author:

Martina Hüllmann

Definition at line 14 of file box.h.

8.4.2 Member Function Documentation

8.4.2.1 double Box::height () const

Returns the length of the box.

Returns:

Length of the box.

Definition at line 22 of file box.cc.

Referenced by Octree::determine_obstacle_max_size().

8.4.2.2 void Box::set_height (double *new_height*)

Sets the height of the box to the given value.

Parameters:

New height of the box.

Definition at line 26 of file box.cc.

8.4.2.3 double Box::depth () const

Returns the depth of the box.

Returns:

Depth of the box.

Definition at line 30 of file box.cc.

Referenced by Octree::determine_obstacle_max_size().

8.4.2.4 void Box::set_depth (double *new_depth*)

Sets the depth of the box to the given value.

Parameters:

New depth of the box.

Definition at line 34 of file box.cc.

8.4.2.5 double Box::width () const

Returns the width of the box.

Returns:

Width of the box.

Definition at line 38 of file box.cc.

Referenced by Octree::determine_obstacle_max_size().

8.4.2.6 void Box::set_width (double *new_width*)

Sets the width of the box to the given value.

Parameters:

New width of the box.

Definition at line 42 of file box.cc.

8.4.2.7 bool Box::contains_point (boost::shared_ptr< Vector3d > *point*) const

Checks whether the given point is contained in the obstacle.

Parameters:

Pointer to vector of point to check whether it's contained in the obstacle.

Definition at line 51 of file box.cc.

8.4.2.8 boost::shared_ptr< WorldObject > Box::clone () const [virtual]

Clones this object and returns a shared ptr to the cloned object. typeid(∗this) == typeid(∗clone)

Returns:

shared ptr to the cloned object

Reimplemented from WorldObject.

Definition at line 46 of file box.cc.

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

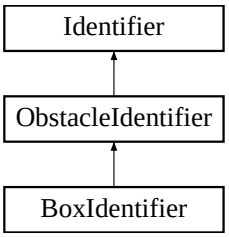
- src/Model/box.h
- src/Model/box.cc

8.5 BoxIdentifier Class Reference

Denote ID's of boxes.

`#include <box_identifier.h>`

Inheritance diagram for BoxIdentifier::



Public Member Functions

- virtual boost::shared_ptr< Identifier > **clone** () const

Protected Member Functions

- **BoxIdentifier** (std::size_t id)

Friends

- class **SimpleWorldFixture**

8.5.1 Detailed Description

Denote ID's of boxes.

Author:

Martina Hüllmann

Definition at line 12 of file box_identifier.h.

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

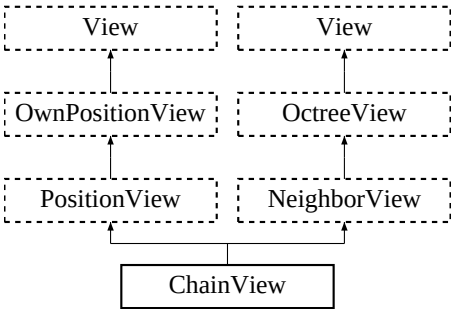
- src/Model/box_identifier.h
- src/Model/box_identifier.cc

8.6 ChainView Class Reference

View model of the robot chain algorithm.

`#include <chain_view.h>`

Inheritance diagram for ChainView::



Public Member Functions

- **ChainView** (unsigned k)

8.6.1 Detailed Description

View model of the robot chain algorithm.

Assigning this class to a Robot corresponds to the robot chain view model, i.e. every Robot can see k neighbor Robots position. Besides this no more information is visible.

See also:

https://wiki.math.uni-paderborn.de/pg-schwarm/StartSeite/AK/Szenarien

Definition at line 25 of file chain_view.h.

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

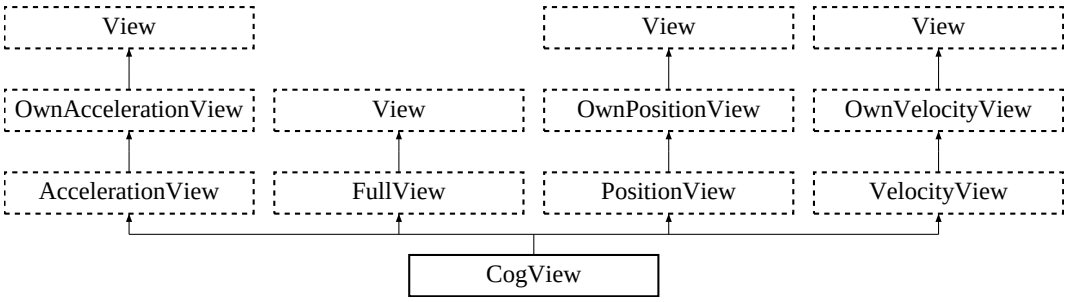
- src/Views/chain_view.h
- src/Views/chain_view.cc

8.7 CogView Class Reference

View model of the classic CenterOfGravity algorithm.

`#include <cog_view.h>`

Inheritance diagram for CogView::



8.7.1 Detailed Description

View model of the classic CenterOfGravity algorithm.

Assigning this class to a Robot corresponds to the COG view model, i.e. every Robot can see every other Robots position, velocity and acceleration. The coordinate-system and id of each Robot is not visible.

See also:

https://wiki.math.uni-paderborn.de/pg-schwarm/StartSeite/AK/Szenarien

Definition at line 27 of file cog_view.h.

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

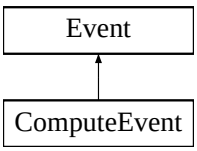
- src/Views/cog_view.h
- src/Views/cog_view.cc

8.8 ComputeEvent Class Reference

A ComputeEvent is an event which causes a subset of the robots to calculate new requests.

`#include <compute_event.h>`

Inheritance diagram for ComputeEvent::



Public Member Functions

- **ComputeEvent** (int time)
- void add_to_robot_subset (boost::shared_ptr< Robot > new_robot)
- const vector< boost::shared_ptr< Robot > > & robot_subset () const
- void add_to_requests (boost::shared_ptr< const Request > new_request)
- const vector< boost::shared_ptr< const Request > > & requests () const

Private Attributes

- vector< boost::shared_ptr< Robot > > robot_subset_
- vector< boost::shared_ptr< const Request > > requests_

8.8.1 Detailed Description

A ComputeEvent is an event which causes a subset of the robots to calculate new requests.

Definition at line 24 of file compute_event.h.

8.8.2 Member Function Documentation

8.8.2.1 void ComputeEvent::add_to_robot_subset (boost::shared_ptr< Robot > *new_robot*)

Adds a new robot to the subset of robots in the event.

Parameters:

A shared pointer to the new robot.

Definition at line 10 of file compute_event.cc.

References robot_subset_.

Referenced by SynchronousASG::get_next_event().

8.8.2.2 const vector< boost::shared_ptr< Robot > > & ComputeEvent::robot_subset () const

Returns a constant reference to the robot subset.

Returns:

A constant reference to the robot subset.

Definition at line 15 of file compute_event.cc.

References robot_subset_.

8.8.2.3 void ComputeEvent::add_to_requests (boost::shared_ptr< const Request > *new_request*)

Adds a new request to the set of requests.

Parameters:

A shared pointer to the new request.

Definition at line 19 of file compute_event.cc.

References requests_.

8.8.2.4 const vector< boost::shared_ptr< const Request > > & ComputeEvent::requests () const

Returns a constant reference to the set of requests.

Returns:

A constant reference to the set of requests.

Definition at line 23 of file compute_event.cc.

References requests_.

Referenced by SynchronousASG::update().

8.8.3 Member Data Documentation

8.8.3.1 vector<boost::shared_ptr<Robot> > ComputeEvent::robot_subset_ [private]

The robot subset for this event.

Definition at line 57 of file compute_event.h.

Referenced by add_to_robot_subset(), and robot_subset().

8.8.3.2 vector<boost::shared_ptr<const Request> > ComputeEvent::requests_ [private]

The set of resulting requests

Definition at line 62 of file compute_event.h.

Referenced by add_to_requests(), and requests().

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

- src/Events/compute_event.h
- src/Events/compute_event.cc

8.9 DistributionGenerator Class Reference

`#include <distribution_generator.h>`

Public Member Functions

- void init_uniform (int min, int max)
- int get_value_uniform ()
- void init_normal (double mean, double sigma)
- double get_value_normal ()
- void init_bernoulli (double probability)
- bool get_value_bernoulli ()
- void init_exponential (double lambda)
- double get_value_exponential ()
- void init_uniform_real (double min, double max)
- double get_value_uniform_real ()
- void init_uniform_on_sphere (int dim)
- std::vector< double > get_value_uniform_on_sphere ()
- Vector3d get_value_uniform_on_sphere_3d ()
- void set_seed (unsigned int seed)
- DistributionGenerator (unsigned int seed)

Private Attributes

- boost::shared_ptr< boost::variate_generator< boost::mt19937 &, boost::uniform_int<> > > **gen_uniform_int_**
- boost::shared_ptr< boost::variate_generator< boost::mt19937 &, boost::normal_distribution<> > > **gen_normal_**
- boost::shared_ptr< boost::variate_generator< boost::mt19937 &, boost::bernoulli_distribution<> > > **gen_bernoulli_**
- boost::shared_ptr< boost::variate_generator< boost::mt19937 &, boost::exponential_distribution<> > > **gen_exponential_**
- boost::shared_ptr< boost::variate_generator< boost::mt19937 &, boost::uniform_real<> > > **gen_uniform_real_**
- boost::shared_ptr< boost::variate_generator< boost::mt19937 &, boost::uniform_on_sphere<> > > **gen_uniform_on_sphere_**
- boost::mt19937 **png_mersenne_**

8.9.1 Detailed Description

This class provides different random number generators for different distributions. Additional generators for specific distributions should be defined here.

Definition at line 31 of file distribution_generator.h.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 DistributionGenerator::DistributionGenerator (unsigned int *seed*)

Constructor

Parameters:

int seed for pseudorandom number generator

Definition at line 105 of file distribution_generator.cc.

References set_seed().

8.9.3 Member Function Documentation

8.9.3.1 void DistributionGenerator::init_uniform (int *min*, int *max*)

Initializes variate_generator for uniform distribution

Parameters:

min int value for integer range

max int value for integer range

Definition at line 23 of file distribution_generator.cc.

8.9.3.2 int DistributionGenerator::get_value_uniform ()

Generates pseudorandom number in according range

Returns:

random int in range

Definition at line 30 of file distribution_generator.cc.

8.9.3.3 void DistributionGenerator::init_normal (double *mean*, double *sigma*)

Initializes variate_generator for normal distribution

$$p(x) = 1/sqrt(2 * pi * sigma) * exp(-(x - mean)^2/(2 * sigma^2))$$

Parameters:

mean double value for normal distribution

sigma double value for normal distribution

Definition at line 34 of file distribution_generator.cc.

8.9.3.4 double DistributionGenerator::get_value_normal ()

Generates pseudorandom number for normal distribution

Returns:

random double value distributed according normal distribution

Definition at line 41 of file distribution_generator.cc.

8.9.3.5 void DistributionGenerator::init_bernoulli (double *probability*)

Initializes variate_generator for bernoulli distribution P(true) = p and P(false) = 1-p

Parameters:

probability double for bernoulli distribution in range [0,1)

Definition at line 45 of file distribution_generator.cc.

8.9.3.6 bool DistributionGenerator::get_value_bernoulli ()

Generates boolen values according to distribution

Returns:

true/false boolean value

Definition at line 51 of file distribution_generator.cc.

8.9.3.7 void DistributionGenerator::init_exponential (double *lambda*)

Initializes variate_generator for exponential distribution

$$p(x) = lambda * exp(-lambda * x)$$

Parameters:

lambda double parameter for distribution, in range [0,1)

Definition at line 55 of file distribution_generator.cc.

8.9.3.8 double DistributionGenerator::get_value_exponential ()

Generates values according to distribution

Returns:

double value

Definition at line 61 of file distribution_generator.cc.

8.9.3.9 void DistributionGenerator::init_uniform_real (double *min*, double *max*)

Initializes variate_generator for normal distribution over the reals

Parameters:

min double value of range

max double value of range

Definition at line 65 of file distribution_generator.cc.

8.9.3.10 double DistributionGenerator::get_value_uniform_real ()

Generates values according to distribution

Returns:

double value

Definition at line 71 of file distribution_generator.cc.

8.9.3.11 void DistributionGenerator::init_uniform_on_sphere (int *dim*)

Initializes variate_generator for uniform distribution on 3-dimensional unit sphere

Parameters:

dimensions

Definition at line 75 of file distribution_generator.cc.

8.9.3.12 std::vector< double > DistributionGenerator::get_value_uniform_on_sphere ()

Generates vector according to distribution

Returns:

vector of doubles

Definition at line 81 of file distribution_generator.cc.

8.9.3.13 Vector3d DistributionGenerator::get_value_uniform_on_sphere_3d ()

Generates Vector3d according to distribution uniform on sphere 3d Needs former initialization by

See also:

init_uniform_on_sphere

Returns:

vector of doubles

Definition at line 85 of file distribution_generator.cc.

8.9.3.14 void DistributionGenerator::set_seed (unsigned int *seed*)

Sets the seed vor PNG

Parameters:

seed must be unsigned int

Definition at line 101 of file distribution_generator.cc.

Referenced by DistributionGenerator().

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

- src/Utilities/distribution_generator.h
- src/Utilities/distribution_generator.cc