

Social Planner

Generated by Doxygen 1.8.11

Contents

1	Todo List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	Agent Class Reference	9
5.1.1	Detailed Description	11
5.1.2	Constructor & Destructor Documentation	11
5.1.2.1	Agent(Vector3d position=Vector3d(), double theta=0, int id=0)	11
5.1.2.2	~Agent()	11
5.1.3	Member Function Documentation	11
5.1.3.1	findNearestNeighbor(std::vector< Agent * > agents)	11
5.1.3.2	getFOVIntersection(Agent *agent)	11
5.1.3.3	getSocialSpace() const	12
5.1.3.4	setSocialSpace(GaussianSpace *socialSpace)	12
5.2	AgentContainer Class Reference	12
5.2.1	Detailed Description	13
5.2.2	Constructor & Destructor Documentation	14

5.2.2.1	AgentContainer()	14
5.2.2.2	AgentContainer(std::vector< Agent * > &agents)	14
5.2.2.3	~AgentContainer()	14
5.2.3	Member Function Documentation	14
5.2.3.1	clearAgents()	14
5.2.3.2	getAgent(unsigned int agentId)	14
5.2.3.3	getAgents()	15
5.2.3.4	pushAgent(Agent *agent)	15
5.2.3.5	removeAgent(unsigned int agentId)	15
5.2.3.6	setAgents(const std::vector< Agent * > &agents)	15
5.3	GridMap::CompaireVCell Struct Reference	15
5.3.1	Detailed Description	16
5.3.2	Member Function Documentation	16
5.3.2.1	operator()(VCell const &a, VCell const &b) const	16
5.4	DrawnObject Class Reference	16
5.4.1	Detailed Description	17
5.4.2	Constructor & Destructor Documentation	17
5.4.2.1	DrawnObject(Vector3d position=Vector3d(), double theta=0)	17
5.4.2.2	~DrawnObject()	18
5.4.3	Member Function Documentation	18
5.4.3.1	pixel_to_real(World *world, Vector3d p)	18
5.4.3.2	real_to_pixel(World *world, Vector3d p)	18
5.5	Formation Class Reference	18
5.5.1	Detailed Description	21
5.5.2	Constructor & Destructor Documentation	21
5.5.2.1	Formation(std::vector< Agent * > agents, int id=0)	21
5.5.2.2	Formation(Agent *agent, int id=0)	21
5.5.2.3	Formation(int id=0)	21
5.5.2.4	~Formation()	21
5.5.3	Member Function Documentation	22

5.5.3.1	getInteractionDirection() const	22
5.5.3.2	getInteractionPosition() const	22
5.5.3.3	getInteractionPotential() const	22
5.5.3.4	getSocialSpace() const	22
5.5.3.5	initAgent(Agent *agent)	22
5.5.3.6	isInFormation(Agent *agent)	23
5.5.3.7	isInFormation(unsigned int agentId)	23
5.5.3.8	pushAgent(Agent *agent)	23
5.5.3.9	removeAgent(unsigned int agentId)	23
5.5.3.10	setInteractionDirection(const Vector3d &interactionDirection)	24
5.5.3.11	setInteractionPosition(const Vector3d &interactionPosition)	24
5.5.3.12	setInteractionPotential(double interactionPotential=0)	24
5.5.3.13	setSocialSpace(OSpace *socialSpace)	24
5.6	GaussianSpace Class Reference	24
5.6.1	Detailed Description	26
5.6.2	Constructor & Destructor Documentation	26
5.6.2.1	GaussianSpace(Agent *agent, int id=0)	26
5.6.2.2	~GaussianSpace()	26
5.6.3	Member Function Documentation	26
5.6.3.1	phi(Vector3d testedRealPoint)	26
5.7	GridCell Class Reference	27
5.7.1	Detailed Description	29
5.7.2	Constructor & Destructor Documentation	29
5.7.2.1	GridCell(double size, double value=0, Vector3d position=Vector3d(), int id=0)	29
5.7.2.2	~GridCell()	29
5.7.3	Member Function Documentation	30
5.7.3.1	getAStarScore() const	30
5.7.3.2	getSize() const	30
5.7.3.3	getValue() const	30
5.7.3.4	isBorderEnabled() const	30

5.7.3.5	<code>isCellSelected()</code> const	30
5.7.3.6	<code>isFrontier()</code> const	31
5.7.3.7	<code>isGoal()</code> const	31
5.7.3.8	<code>isInfoEnabled()</code> const	31
5.7.3.9	<code>isProcessed()</code> const	31
5.7.3.10	<code>isStart()</code> const	31
5.7.3.11	<code>setAStarScore(int starScore)</code>	31
5.7.3.12	<code>setBorderEnabled(bool borderEnabled=true)</code>	32
5.7.3.13	<code>setCellSelected(bool cellSelected=true)</code>	32
5.7.3.14	<code>setFrontier(bool frontier=true)</code>	32
5.7.3.15	<code>setGoal(bool goal=true)</code>	32
5.7.3.16	<code>setInfoEnabled(bool infoEnabled=true)</code>	32
5.7.3.17	<code>setProcessed(bool processed=true)</code>	33
5.7.3.18	<code>setSize(double size)</code>	33
5.7.3.19	<code>setStart(bool start=true)</code>	33
5.7.3.20	<code>setValue(double value)</code>	33
5.8	GridMap Class Reference	33
5.8.1	Detailed Description	36
5.8.2	Constructor & Destructor Documentation	36
5.8.2.1	<code>GridMap(World *world, Population *pop, double resolution=0.1)</code>	36
5.8.2.2	<code>~GridMap()</code>	37
5.8.3	Member Function Documentation	37
5.8.3.1	<code>constructPath()</code>	37
5.8.3.2	<code>findPath(GridCell *startCell, GridCell *endCell)</code>	37
5.8.3.3	<code>getCell(unsigned int cellId)</code>	37
5.8.3.4	<code>getCell(double x, double y)</code>	37
5.8.3.5	<code>isBorderEnabled()</code> const	38
5.8.3.6	<code>isGroupSpaceEnabled()</code> const	38
5.8.3.7	<code>isPersonalSpaceEnabled()</code> const	38
5.8.3.8	<code>neighbors(GridCell *cell, bool allowDiagonalMove=true)</code>	38

5.8.3.9	pathFinderNextStep()	39
5.8.3.10	setBorderEnabled(bool borderEnabled=true)	39
5.8.3.11	setGroupSpaceEnabled(bool groupSpaceEnabled=true)	39
5.8.3.12	setInfoEnabled(bool infoEnabled=false)	39
5.8.3.13	setPersonalSpaceEnabled(bool personalSpaceEnabled=true)	39
5.9	GroupDetector Class Reference	40
5.9.1	Detailed Description	41
5.9.2	Constructor & Destructor Documentation	41
5.9.2.1	GroupDetector(Population *pop)	41
5.9.2.2	~GroupDetector()	41
5.10	GroupSocialSpace Class Reference	41
5.10.1	Detailed Description	43
5.10.2	Constructor & Destructor Documentation	43
5.10.2.1	GroupSocialSpace(int id=0)	43
5.10.2.2	GroupSocialSpace(std::vector< Agent * > &agents, int id=0)	43
5.10.2.3	~GroupSocialSpace()	43
5.11	Gui Class Reference	43
5.11.1	Detailed Description	44
5.12	IdentifiedObject Class Reference	44
5.12.1	Detailed Description	45
5.12.2	Constructor & Destructor Documentation	45
5.12.2.1	IdentifiedObject(unsigned int id=0)	45
5.12.2.2	~IdentifiedObject()	45
5.12.3	Member Function Documentation	45
5.12.3.1	getId() const	45
5.12.3.2	setId(unsigned int id)	45
5.13	LocalizedObject Class Reference	46
5.13.1	Detailed Description	47
5.13.2	Constructor & Destructor Documentation	47
5.13.2.1	LocalizedObject(Vector3d position=Vector3d(), double theta=0)	47

5.13.2.2	<code>~LocalizedObject()</code>	47
5.13.3	Member Function Documentation	47
5.13.3.1	<code>getDirection()</code>	47
5.13.3.2	<code>getPosition() const</code>	48
5.13.3.3	<code>getTheta() const</code>	48
5.13.3.4	<code>getX() const</code>	48
5.13.3.5	<code>getY() const</code>	48
5.13.3.6	<code>setPosition(Vector3d position)</code>	48
5.13.3.7	<code>setTheta(double theta)</code>	48
5.13.3.8	<code>setX(double x)</code>	49
5.13.3.9	<code>setY(double y)</code>	49
5.14	ofApp Class Reference	49
5.14.1	Detailed Description	51
5.15	OSpace Class Reference	51
5.15.1	Detailed Description	53
5.15.2	Constructor & Destructor Documentation	53
5.15.2.1	<code>OSpace(int id=0)</code>	53
5.15.2.2	<code>OSpace(std::vector< Agent * > &agents, int id=0)</code>	53
5.15.2.3	<code>~OSpace()</code>	54
5.15.3	Member Function Documentation	54
5.15.3.1	<code>computeCovarMatrix()</code>	54
5.15.3.2	<code>getCenter() const</code>	54
5.15.3.3	<code>getgCenter() const</code>	54
5.15.3.4	<code>less(Vector3d a, Vector3d b)</code>	54
5.15.3.5	<code>phi(Vector3d testedPoint)</code>	55
5.15.3.6	<code>setCenter(const Vector3d &center)</code>	55
5.15.3.7	<code>setgCenter(const Vector3d &gCenter)</code>	55
5.16	PersonnalSocialSpace Class Reference	56
5.16.1	Detailed Description	57
5.16.2	Constructor & Destructor Documentation	57

5.16.2.1	<code>PersonnalSocialSpace(Agent *agent, int id=0)</code>	57
5.16.2.2	<code>~PersonnalSocialSpace()</code>	58
5.17	Population Class Reference	58
5.17.1	Detailed Description	60
5.17.2	Constructor & Destructor Documentation	60
5.17.2.1	<code>Population(std::vector< Agent * > agents, Vector3d position=Vector3d(), int id=0)</code>	60
5.17.2.2	<code>Population(Vector3d position=Vector3d(), int id=0)</code>	60
5.17.2.3	<code>~Population()</code>	60
5.17.3	Member Function Documentation	60
5.17.3.1	<code>clear()</code>	60
5.17.3.2	<code>clearFormations()</code>	61
5.17.3.3	<code>getFormations() const</code>	61
5.17.3.4	<code>getHighestFormationInteractionPotential()</code>	61
5.17.3.5	<code>getRelatedFormation(unsigned int agentId)</code>	61
5.17.3.6	<code>getRelatedFormation(Agent *a)</code>	61
5.17.3.7	<code>isGrouped(Agent *agent)</code>	61
5.17.3.8	<code>isGrouped(unsigned int agentId)</code>	62
5.17.3.9	<code>pushFormation(Formation *formation)</code>	62
5.17.3.10	<code>removeFormation(unsigned int formationId)</code>	62
5.17.3.11	<code>setFormations(const std::vector< Formation * > &formations)</code>	62
5.18	PopulationManager Class Reference	63
5.19	Robot Class Reference	65
5.19.1	Detailed Description	67
5.19.2	Constructor & Destructor Documentation	67
5.19.2.1	<code>Robot(Vector3d position=Vector3d(0, 0, 0), double theta=0)</code>	67
5.19.2.2	<code>~Robot()</code>	68
5.19.3	Member Function Documentation	68
5.19.3.1	<code>getGoal() const</code>	68
5.19.3.2	<code>getPath() const</code>	68
5.19.3.3	<code>setGoal(GridCell *goal)</code>	68

5.19.3.4	setPath(std::vector< GridCell * > path)	68
5.20	SocialPlanner Class Reference	69
5.20.1	Detailed Description	70
5.20.2	Constructor & Destructor Documentation	70
5.20.2.1	SocialPlanner(PopulationManager *popManager, Robot *robot)	70
5.20.2.2	~SocialPlanner()	71
5.20.3	Member Function Documentation	71
5.20.3.1	disengage()	71
5.20.3.2	engage()	71
5.20.3.3	getManager() const	71
5.20.3.4	getRobot() const	71
5.20.3.5	interact()	72
5.20.3.6	seek_interaction()	72
5.20.3.7	update()	72
5.21	SocialSpace Class Reference	72
5.21.1	Detailed Description	73
5.21.2	Constructor & Destructor Documentation	73
5.21.2.1	SocialSpace(int id=0)	73
5.21.2.2	~SocialSpace()	73
5.22	UDPServer Class Reference	74
5.22.1	Detailed Description	75
5.22.2	Constructor & Destructor Documentation	75
5.22.2.1	UDPServer(int port, SocialPlanner *sPlanner)	75
5.22.2.2	~UDPServer()	76
5.22.3	Member Function Documentation	76
5.22.3.1	do_read()	76
5.22.3.2	do_send(uint8_t *sendBuffer, int sendBuffer_size)	76
5.22.3.3	parse()	76
5.22.3.4	parse_frame0()	77
5.22.3.5	send_frame0()	77

5.22.3.6	send_frame1()	77
5.22.3.7	send_frame2()	77
5.22.3.8	send_frame3()	77
5.22.3.9	spawn()	77
5.22.3.10	updateOrPushAgent(int id, float x, float y, float z, float theta)	77
5.23	World Class Reference	78
5.23.1	Detailed Description	80
5.23.2	Constructor & Destructor Documentation	80
5.23.2.1	World(double width, double height, int widthView, int heightView, Vector3d position=Vector3d(), double theta=0)	80
5.23.2.2	~World()	80
5.23.3	Member Function Documentation	80
5.23.3.1	getHeight() const	80
5.23.3.2	getHeightView() const	80
5.23.3.3	getWidth() const	81
5.23.3.4	getWidthView() const	81
5.23.3.5	setHeight(double height)	81
5.23.3.6	setHeightView(double heightView)	81
5.23.3.7	setWidth(double width)	81
5.23.3.8	setWidthView(double widthView)	81
6	File Documentation	83
6.1	src/agentManagement/Agent.cpp File Reference	83
6.1.1	Detailed Description	83
6.2	src/agentManagement/Agent.h File Reference	84
6.2.1	Detailed Description	85
6.3	src/agentManagement/Formation.cpp File Reference	85
6.3.1	Detailed Description	86
6.4	src/agentManagement/Formation.h File Reference	86
6.4.1	Detailed Description	87
6.5	src/agentManagement/Population.cpp File Reference	88

6.5.1	Detailed Description	88
6.6	src/agentManagement/Population.h File Reference	88
6.6.1	Detailed Description	89
6.7	src/agentManagement/Robot.h File Reference	90
6.7.1	Detailed Description	91
6.8	src/genericType/AgentContainer.cpp File Reference	92
6.8.1	Detailed Description	92
6.9	src/genericType/AgentContainer.h File Reference	93
6.9.1	Detailed Description	94
6.10	src/genericType/DrawnObject.cpp File Reference	95
6.10.1	Detailed Description	95
6.11	src/genericType/DrawnObject.h File Reference	95
6.11.1	Detailed Description	96
6.12	src/genericType/IdentifiedObject.cpp File Reference	97
6.12.1	Detailed Description	97
6.13	src/genericType/IdentifiedObject.h File Reference	97
6.13.1	Detailed Description	98
6.14	src/genericType/LocalizedObject.cpp File Reference	98
6.14.1	Detailed Description	98
6.15	src/genericType/LocalizedObject.h File Reference	99
6.15.1	Detailed Description	99
6.16	src/socialProcessing/GroupDetector.cpp File Reference	100
6.16.1	Detailed Description	100
6.17	src/socialProcessing/GroupDetector.h File Reference	101
6.17.1	Detailed Description	102
6.18	src/socialProcessing/SocialPlanner.cpp File Reference	102
6.18.1	Detailed Description	103
6.19	src/socialProcessing/SocialPlanner.h File Reference	103
6.19.1	Detailed Description	104
6.20	src/socialSpace/GaussianSpace.cpp File Reference	104

6.20.1 Detailed Description	105
6.21 src/socialSpace/GaussianSpace.h File Reference	105
6.21.1 Detailed Description	107
6.22 src/socialSpace/GroupSocialSpace.h File Reference	107
6.22.1 Detailed Description	108
6.23 src/socialSpace/OSpace.cpp File Reference	109
6.23.1 Detailed Description	109
6.24 src/socialSpace/OSpace.h File Reference	110
6.24.1 Detailed Description	111
6.25 src/socialSpace/PersonnalSocialSpace.cpp File Reference	112
6.25.1 Detailed Description	112
6.26 src/socialSpace/PersonnalSocialSpace.h File Reference	113
6.26.1 Detailed Description	114
6.27 src/socialSpace/SocialSpace.cpp File Reference	114
6.27.1 Detailed Description	114
6.28 src/socialSpace/SocialSpace.h File Reference	115
6.28.1 Detailed Description	116
6.29 src/UDPServer.h File Reference	116
6.29.1 Detailed Description	117
6.30 src/worldRepresentation/GridCell.cpp File Reference	117
6.30.1 Detailed Description	118
6.31 src/worldRepresentation/GridCell.h File Reference	118
6.31.1 Detailed Description	119
6.32 src/worldRepresentation/GridMap.cpp File Reference	120
6.32.1 Detailed Description	120
6.32.2 Function Documentation	120
6.32.2.1 heuristicDiagonalCostEstimate(GridCell *start, GridCell *end)	120
6.32.2.2 heuristicManhattanCostEstimate(GridCell *start, GridCell *end)	121
6.33 src/worldRepresentation/GridMap.h File Reference	121
6.33.1 Detailed Description	122
6.34 src/worldRepresentation/World.h File Reference	123
6.34.1 Detailed Description	124

Chapter 1

Todo List

Member `Agent::getFOVIntersection (Agent *agent)`

Take parallel `Agent` into account, actually return NULL if Agents are parallel

Class `GroupDetector`

This class should be an interface and actual logic should be implemented has `BasicGroupDetector`

Class `SocialPlanner`

This class should be an interface to implement different behavior for the robot

Member `SocialPlanner::disengage ()`

Execute some social signals for disengagement

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

AgentContainer	12
Formation	18
GroupSocialSpace	41
OSpace	51
Population	58
GridMap::CompaireVCell	15
GroupDetector	40
Gui	43
IdentifiedObject	44
Agent	9
Formation	18
GridCell	27
GridMap	33
Population	58
Robot	65
SocialSpace	72
GroupSocialSpace	41
PersonnalSocialSpace	56
GaussianSpace	24
LocalizedObject	46
DrawnObject	16
Agent	9
Formation	18
GridCell	27
GridMap	33
OSpace	51
Population	58
PopulationManager	63
Robot	65
World	78
ofBaseApp	
ofApp	49
SocialPlanner	69
UDPServer	74

Chapter 3

Class Index

3.1 Class List

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

Agent	This class represent the Agents	9
AgentContainer	This class is an interface for class that contains multiples Agents	12
GridMap::CompaireVCell	Define an operator for GridCell and associated gScore comparison for A* algorithm	15
DrawnObject	This class is an interface for class that are drawn on OFX gui	16
Formation	This class represent the social Formation	18
GaussianSpace	This class is an implementation of the PersonnalSocialSpace	24
GridCell	This class represent a cell in the GridMap	27
GridMap	This class manage the 2D GridMap computed from the Agents SocialSpace	33
GroupDetector	This class is dedicated to process every Agents in the Population and create every Formation	40
GroupSocialSpace	This class is an interface to implement representation of a GroupSocialSpace	41
Gui	Graphical User Interface	43
IdentifiedObject	This class is an interface for class that need a unique identifier	44
LocalizedObject	This class is an interface for object that are localized in real World	46
ofApp	The Openframeworks main class	49
OSpace	This class is an implementation of the GroupSocialSpace	51
PersonnalSocialSpace	This class is an interface to implement representation of a PersonnalSocialSpace	56
Population	This class represent Population around the Robot	58
PopulationManager	63

Robot	This class represent the Robot	65
SocialPlanner	This class is a state machine controlling the Robot behavior	69
SocialSpace	This class is an abstract class for representing SocialSpace	72
UDPServer	This class manage the UDP Server sending computed data to a visualization software and receiving Agent data from other sensor sources	74
World	This class represent the main frame coordinates and its projection in pixels	78

Chapter 4

File Index

4.1 File List

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

src/ config.h	??
src/ Gui.h	??
src/ ofApp.h	??
src/ UDPServer.h	116
src/ utils.h	??
src/agentManagement/ Agent.cpp	83
src/agentManagement/ Agent.h	84
src/agentManagement/ Formation.cpp	85
src/agentManagement/ Formation.h	86
src/agentManagement/ Population.cpp	88
src/agentManagement/ Population.h	88
src/agentManagement/ PopulationManager.h	??
src/agentManagement/ Robot.h	90
src/genericType/ AgentContainer.cpp	92
src/genericType/ AgentContainer.h	93
src/genericType/ DrawnObject.cpp	95
src/genericType/ DrawnObject.h	95
src/genericType/ IdentifiedObject.cpp	97
src/genericType/ IdentifiedObject.h	97
src/genericType/ LocalizedObject.cpp	98
src/genericType/ LocalizedObject.h	99
src/socialProcessing/ GroupDetector.cpp	100
src/socialProcessing/ GroupDetector.h	101
src/socialProcessing/ SocialPlanner.cpp	102
src/socialProcessing/ SocialPlanner.h	103
src/socialSpace/ GaussianSpace.cpp	104
src/socialSpace/ GaussianSpace.h	105
src/socialSpace/ GroupSocialSpace.h	107
src/socialSpace/ OSpace.cpp	109
src/socialSpace/ OSpace.h	110
src/socialSpace/ PersonnalSocialSpace.cpp	112
src/socialSpace/ PersonnalSocialSpace.h	113
src/socialSpace/ SocialSpace.cpp	114
src/socialSpace/ SocialSpace.h	115
src/worldRepresentation/ GridCell.cpp	117

src/worldRepresentation/ GridCell.h	118
src/worldRepresentation/ GridMap.cpp	120
src/worldRepresentation/ GridMap.h	121
src/worldRepresentation/ World.h	123

Chapter 5

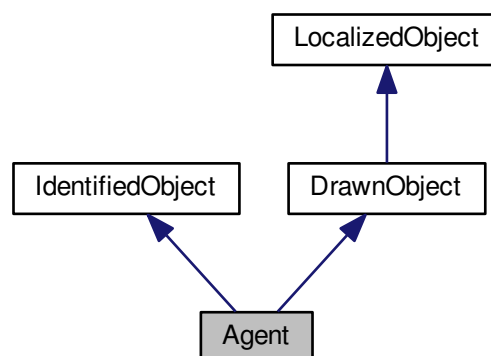
Class Documentation

5.1 Agent Class Reference

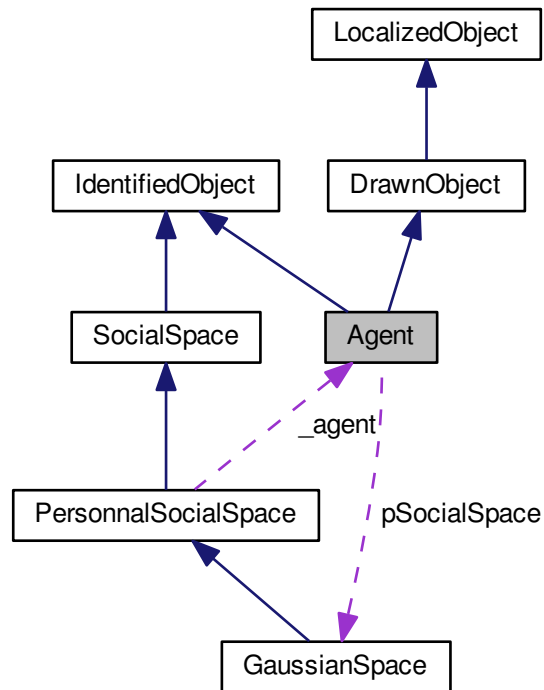
This class represent the Agents.

```
#include <Agent.h>
```

Inheritance diagram for Agent:



Collaboration diagram for Agent:



Public Member Functions

- **Agent** (Vector3d **position**=Vector3d(), double **theta**=0, int **id**=0)
Constructor.
- **~Agent** ()
Destructor.
- **Agent * findNearestNeighbor** (std::vector< **Agent** * > agents)
Find the nearest neighbor available in the list passed in parameters.
- Vector3d * **getFOVIntersection** (**Agent** *agent)
*Find the intersection point in the field of view with the other **Agent** passed in parameter.*
- **GaussianSpace * getSocialSpace** () const
Simple getter.
- void **setSocialSpace** (**GaussianSpace** *socialSpace)
Simple setter.

Protected Attributes

- **GaussianSpace * pSocialSpace**
*PersonnalSocialSpace related to the **Agent**.*

5.1.1 Detailed Description

This class represent the Agents.

This class manage the Agents detected around the robot

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Agent::Agent (Vector3d *position* = Vector3d(), double *theta* = 0, int *id* = 0)

Constructor.

Constructor of the [Agent](#) class, initialize a [PersonnalSocialSpace](#)

Parameters

<i>position</i>	: Initial position of the Agent in World
<i>theta</i>	: Initial angle of the Agent
<i>id</i>	: Unique identifier of the Agent

5.1.2.2 Agent::~Agent ()

Destructor.

Destructor of the [Agent](#) class, destroy the related [PersonnalSocialSpace](#)

5.1.3 Member Function Documentation

5.1.3.1 Agent * Agent::findNearestNeighbor (std::vector< Agent * > *agents*)

Find the nearest neighbor available in the list passed in parameters.

Parameters

<i>agents</i>	: A list of Agents
---------------	--------------------

Returns

The nearest [Agent](#) available in agents

5.1.3.2 Vector3d * Agent::getFOVIntersection (Agent * *agent*)

Find the intersection point in the field of view with the other [Agent](#) passed in parameter.

Parameters

<i>agent</i>	: The targeted Agent
--------------	--------------------------------------

Returns

The intersection point in the field of view of both [Agent](#), returns NULL if there is no intersection

Todo Take parallel [Agent](#) into account, actually return NULL if Agents are parallel

5.1.3.3 `GaussianSpace * Agent::getSocialSpace () const`

Simple getter.

Returns

PersonalSocialSpace

5.1.3.4 `void Agent::setSocialSpace (GaussianSpace * socialSpace)`

Simple setter.

Parameters

<i>socialSpace</i>	
--------------------	--

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

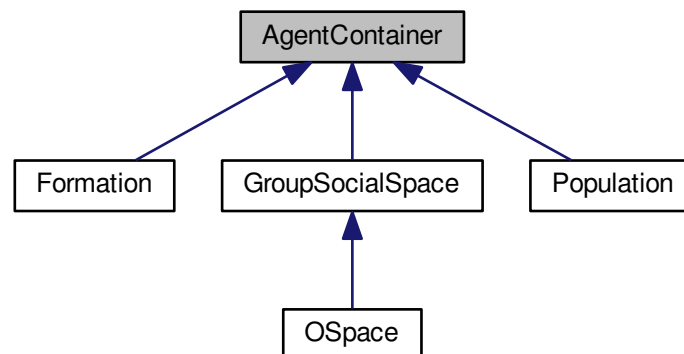
- [src/agentManagement/Agent.h](#)
- [src/agentManagement/Agent.cpp](#)

5.2 AgentContainer Class Reference

This class is an interface for class that contains multiples Agents.

```
#include <AgentContainer.h>
```

Inheritance diagram for AgentContainer:



Public Member Functions

- [AgentContainer](#) ()
Constructor.
- [AgentContainer](#) (std::vector< [Agent](#) * > &agents)
Constructor.
- virtual [~AgentContainer](#) ()
Destructor.
- void [pushAgent](#) ([Agent](#) *agent)
Add an [Agent](#) to the container.
- int [removeAgent](#) (unsigned int agentId)
Remove an [Agent](#) by id from the container.
- void [clearAgents](#) ()
Clear the container.
- [Agent](#) * [getAgent](#) (unsigned int agentId)
Get an [Agent](#) by id from the container.
- std::vector< [Agent](#) * > [getAgents](#) ()
Simple getter.
- void [setAgents](#) (const std::vector< [Agent](#) * > &agents)
Simple setter.

Protected Attributes

- std::vector< [Agent](#) * > [_agents](#)
The [Agent](#) container.

5.2.1 Detailed Description

This class is an interface for class that contains multiples Agents.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AgentContainer::AgentContainer ()

Constructor.

Constructor of the [AgentContainer](#) class, initialize an empty container

5.2.2.2 AgentContainer::AgentContainer (std::vector< Agent * > & agents)

Constructor.

Constructor of the [AgentContainer](#) class, initialize the container with agents

Parameters

<i>agents</i>	: The Agents present in the container
---------------	---------------------------------------

5.2.2.3 AgentContainer::~~AgentContainer () [virtual]

Destructor.

Destructor of the [AgentContainer](#) class

5.2.3 Member Function Documentation

5.2.3.1 void AgentContainer::clearAgents ()

Clear the container.

Clear the container, this function doesn't call the [Agent](#) destructor

5.2.3.2 Agent * AgentContainer::getAgent (unsigned int agentId)

Get an [Agent](#) by id from the container.

Parameters

<i>agentId</i>	: Agent id to get from the container
----------------	--

Returns

The [Agent](#), null otherwise

5.2.3.3 `std::vector< Agent * > AgentContainer::getAgents ()`

Simple getter.

Returns

Agents

5.2.3.4 `void AgentContainer::pushAgent (Agent * agent)`

Add an [Agent](#) to the container.

Parameters

<i>agent</i>	: The Agent to add
--------------	------------------------------------

5.2.3.5 `int AgentContainer::removeAgent (unsigned int agentId)`

Remove an [Agent](#) by id from the container.

Parameters

<i>agentId</i>	: The Agent id to remove
----------------	--

Returns

0 on success, -1 if [Agent](#) id is not found

5.2.3.6 `void AgentContainer::setAgents (const std::vector< Agent * > & agents)`

Simple setter.

Parameters

<i>agents</i>	
---------------	--

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

- `src/genericType/AgentContainer.h`
- `src/genericType/AgentContainer.cpp`

5.3 GridMap::CompaireVCell Struct Reference

Define an operator for [GridCell](#) and associated gScore comparison for A* algorithm.

```
#include <GridMap.h>
```

Public Member Functions

- `bool operator() (VCell const &a, VCell const &b) const`

5.3.1 Detailed Description

Define an operator for `GridCell` and associated gScore comparison for A* algorithm.

5.3.2 Member Function Documentation

5.3.2.1 `bool GridMap::CompaireVCell::operator() (VCell const & a, VCell const & b) const` `[inline]`

Parameters

<i>a</i>	: The VCell compared
<i>b</i>	: The other VCell compared

Returns

- 1 if VCell have a greater score than the VCell b

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

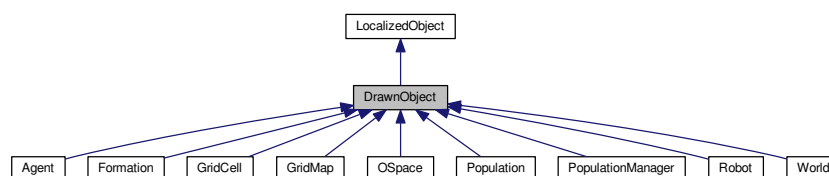
- `src/worldRepresentation/GridMap.h`

5.4 DrawnObject Class Reference

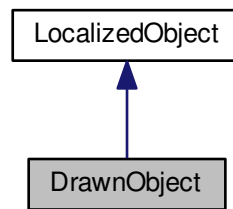
This class is an interface for class that are drawn on OFX gui.

```
#include <DrawnObject.h>
```

Inheritance diagram for DrawnObject:



Collaboration diagram for DrawnObject:



Public Member Functions

- `DrawnObject` (`Vector3d position=Vector3d()`, `double theta=0`)
Constructor.
- `virtual ~DrawnObject ()`
Destructor.
- `Vector3d real_to_pixel` (`World *world`, `Vector3d p`)
The projection function from real to pixel coordinates.
- `Vector3d pixel_to_real` (`World *world`, `Vector3d p`)
The projection function from pixel to real coordinates.

Additional Inherited Members

5.4.1 Detailed Description

This class is an interface for class that are drawn on OFX gui.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 `DrawnObject::DrawnObject (Vector3d position = Vector3d() , double theta = 0)`

Constructor.

Constructor of the `DrawnObject` class

Parameters

<i>position</i>	: The initial position of the object
<i>theta</i>	: The initial rotation of the object

5.4.2.2 DrawnObject::~~DrawnObject () [virtual]

Destructor.

Destructor of the [DrawnObject](#) class

5.4.3 Member Function Documentation

5.4.3.1 Vector3d DrawnObject::pixel_to_real (World * world, Vector3d p)

The projection function from pixel to real coordinates.

Parameters

<i>world</i>	: The World main coordinates frame
<i>p</i>	: The vector that will be projected

Returns

The projected vector in real coordinates

5.4.3.2 Vector3d DrawnObject::real_to_pixel (World * world, Vector3d p)

The projection function from real to pixel coordinates.

Parameters

<i>world</i>	: The World main coordinates frame
<i>p</i>	: The vector that will be projected

Returns

The projected vector in pixel coordinates

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

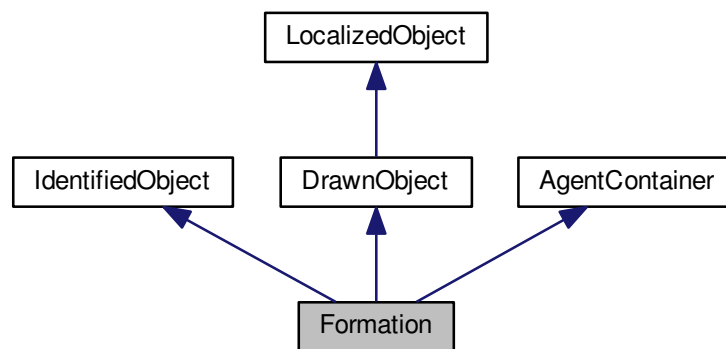
- [src/genericType/DrawnObject.h](#)
- [src/genericType/DrawnObject.cpp](#)

5.5 Formation Class Reference

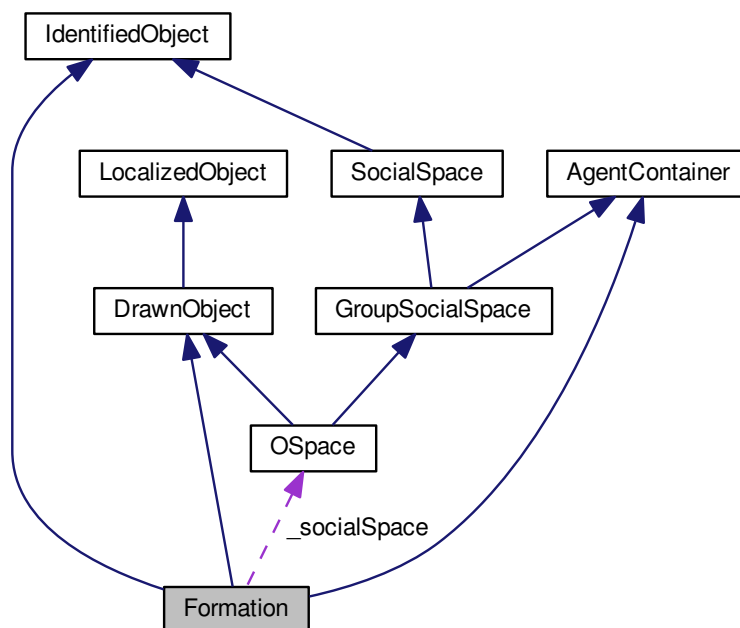
This class represent the social [Formation](#).

```
#include <Formation.h>
```


Inheritance diagram for Formation:



Collaboration diagram for Formation:



Public Member Functions

- **Formation** (std::vector< **Agent** * > agents, int id=0)
Constructor.
- **Formation** (**Agent** *agent, int id=0)

- Constructor.
- **Formation** (int id=0)
 - Constructor.
- **~Formation** ()
 - Destructor.
- void **computeInteractionPotential** ()
 - Compute the interaction potential of the formation based on the mean interpersonal distance and distance from the **Formation** center.
- void **findInteractionPosition** ()
 - Compute the interaction position from the robot to adopt in order to take part in the **Formation**.
- void **update** ()
 - Update **Formation** related data by calling **GroupSocialSpace** update function, computeInteractionPotential and findInteractionPosition.
- void **pushAgent** (**Agent** *agent)
 - Add an **Agent** to the **Formation** and update/create the corresponding **GroupSocialSpace**.
- void **removeAgent** (unsigned int agentId)
 - Remove an **Agent** by id from the **Formation** and the corresponding **GroupSocialSpace**.
- int **isInFormation** (**Agent** *agent)
 - Check if the **Agent** passed in parameter is taking part in the **Formation** Alias to isInFormation function by **Agent** id.
- int **isInFormation** (unsigned int agentId)
 - Check if the **Agent** passed in parameter is taking part in the **Formation**.
- std::vector< **Agent** * > **initAgent** (**Agent** *agent)
 - Initialization function for **Formation** Constructor with only one **Agent**.
- **OSpace** * **getSocialSpace** () const
 - Simple getter.
- void **setSocialSpace** (**OSpace** *socialSpace)
 - Simple setter.
- double **getInteractionPotential** () const
 - Simple getter.
- void **setInteractionPotential** (double interactionPotential=0)
 - Simple setter.
- const Vector3d & **getInteractionPosition** () const
 - Simple getter.
- void **setInteractionPosition** (const Vector3d &interactionPosition)
 - Simpler setter.
- const Vector3d & **getInteractionDirection** () const
 - Simpler getter.
- void **setInteractionDirection** (const Vector3d &interactionDirection)
 - Simple setter.

Protected Attributes

- **OSpace** * **_socialSpace**
 - GroupSocialSpace** related to the **Formation**.
- double **interactionPotential** = 0
 - Estimated interaction potential of the **Formation**.
- Vector3d **interactionPosition**
 - Estimated interaction position for the **Robot** to take part in the **Formation**.
- Vector3d **interactionDirection**
 - Estimated interaction direction for the **Robot** to take part in the **Formation**.
- std::vector< Vector3d > **agentDir_ospace**
 - Vector3d of Agents direction to the **Formation** center.

5.5.1 Detailed Description

This class represent the social [Formation](#).

This class manage the social Formations created by the Agents

5.5.2 Constructor & Destructor Documentation

5.5.2.1 `Formation::Formation (std::vector< Agent * > agents, int id = 0)`

Constructor.

Constructor of the [Formation](#) class, initialize a [GroupSocialSpace](#)

Parameters

<i>agents</i>	: Agents that are part of the Formation
<i>id</i>	: Unique identifier of the Formation

5.5.2.2 `Formation::Formation (Agent * agent, int id = 0)`

Constructor.

Constructor of the [Formation](#) class, this constructor doesn't initialize any [GroupSocialSpace](#)

Parameters

<i>agent</i>	: Agent that is part of the Formation
<i>id</i>	: Unique identifier of the Formation

5.5.2.3 `Formation::Formation (int id = 0)`

Constructor.

Constructor of the [Formation](#) class, creates an empty formation. This constructor doesn't initialize any [Group↔SocialSpace](#)

Parameters

<i>id</i>	: Unique identifier of the Formation
-----------	--

5.5.2.4 `Formation::~~Formation ()`

Destructor.

Destructor of the [Formation](#) class, destroy the related [GroupSocialSpace](#) if defined

5.5.3 Member Function Documentation

5.5.3.1 `const Vector3d & Formation::getInteractionDirection () const`

Simpler getter.

Returns

InteractionDirection

5.5.3.2 `const Vector3d & Formation::getInteractionPosition () const`

Simple getter.

Returns

InteractionPosition

5.5.3.3 `double Formation::getInteractionPotential () const`

Simple getter.

Returns

InteractionPotential

5.5.3.4 `OSpace * Formation::getSocialSpace () const`

Simple getter.

Returns

[GroupSocialSpace](#)

5.5.3.5 `std::vector< Agent * > Formation::initAgent (Agent * agent)`

Initialization function for [Formation](#) Constructor with only one [Agent](#).

Parameters

<i>agent</i>	: Agent initialized for the Formation
--------------	---

Returns

A vector containing the [Agent](#) passed in parameter.

5.5.3.6 int Formation::isInFormation (Agent * agent)

Check if the [Agent](#) passed in parameter is taking part in the [Formation](#) Alias to isInFormation function by [Agent](#) id.

Parameters

<i>agent</i>	: Agent pointer to check
--------------	--

Returns

1 if the [Agent](#) is taking part in the [Formation](#), 0 otherwise

5.5.3.7 int Formation::isInFormation (unsigned int agentId)

Check if the [Agent](#) passed in parameter is taking part in the [Formation](#).

Parameters

<i>agentId</i>	: Agent id to check
----------------	-------------------------------------

Returns

1 if the [Agent](#) is taking part in the [Formation](#), 0 otherwise

5.5.3.8 void Formation::pushAgent (Agent * agent)

Add an [Agent](#) to the [Formation](#) and update/create the corresponding [GroupSocialSpace](#).

Parameters

<i>agent</i>	: Agent pointer to add to the formation
--------------	---

5.5.3.9 void Formation::removeAgent (unsigned int agentId)

Remove an [Agent](#) by id from the [Formation](#) and the corresponding [GroupSocialSpace](#).

Parameters

<i>agentId</i>	: Agent id to remove from the Formation
----------------	---

5.5.3.10 void Formation::setInteractionDirection (const Vector3d & *interactionDirection*)

Simple setter.

Parameters

<i>interactionDirection</i>	
-----------------------------	--

5.5.3.11 void Formation::setInteractionPosition (const Vector3d & *interactionPosition*)

Simpler setter.

Parameters

<i>interactionPosition</i>	
----------------------------	--

5.5.3.12 void Formation::setInteractionPotential (double *interactionPotential* = 0)

Simple setter.

Parameters

<i>interactionPotential</i>	
-----------------------------	--

5.5.3.13 void Formation::setSocialSpace (OSpace * *socialSpace*)

Simple setter.

Parameters

<i>socialSpace</i>	
--------------------	--

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

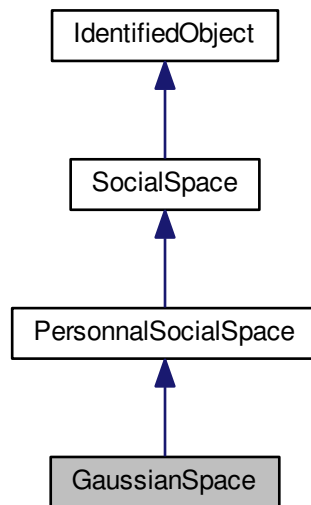
- [src/agentManagement/Formation.h](#)
- [src/agentManagement/Formation.cpp](#)

5.6 GaussianSpace Class Reference

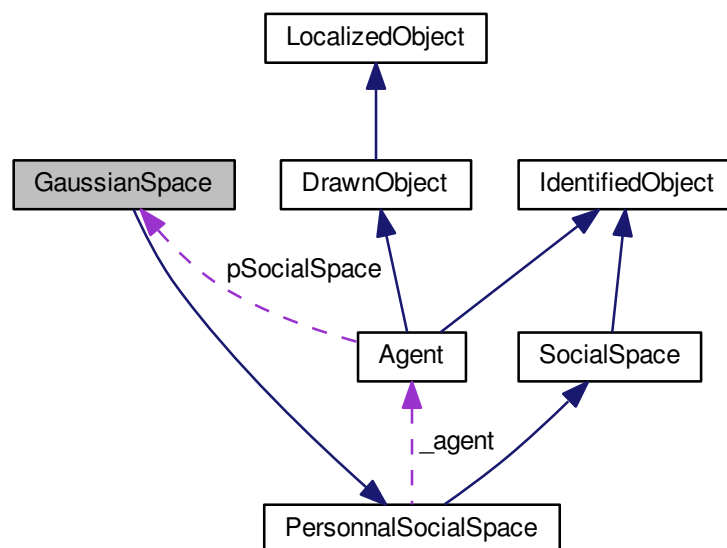
This class is an implementation of the [PersonnalSocialSpace](#).

```
#include <GaussianSpace.h>
```

Inheritance diagram for GaussianSpace:



Collaboration diagram for GaussianSpace:



Public Member Functions

- [GaussianSpace](#) ([Agent](#) *agent, int id=0)

Constructor.

- virtual [~GaussianSpace](#) ()

Destructor.

- double [phi](#) (Vector3d testedRealPoint)

Compute the value of the [GaussianSpace](#) at a given point in space.

Additional Inherited Members

5.6.1 Detailed Description

This class is an implementation of the [PersonnalSocialSpace](#).

This class is an implementation of the [PersonnalSocialSpace](#) represented by a 2D gaussian mixture model model based on interpersonal distances.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 [GaussianSpace::GaussianSpace](#) ([Agent](#) * *agent*, int *id* = 0)

Constructor.

Constructor of the [GaussianSpace](#) class

Parameters

<i>agent</i>	: The Agent related to the GaussianSpace
<i>id</i>	: The unique identifier of the GaussianSpace

5.6.2.2 [GaussianSpace::~~GaussianSpace](#) () [virtual]

Destructor.

Destructor of the [GaussianSpace](#) class

5.6.3 Member Function Documentation

5.6.3.1 double [GaussianSpace::phi](#) ([Vector3d](#) *testedRealPoint*)

Compute the value of the [GaussianSpace](#) at a given point in space.

Parameters

<i>testedRealPoint</i>	: The coordinates of the point in the real frame coordinates World
------------------------	--

Returns

The value of the [GaussianSpace](#) at the given point

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

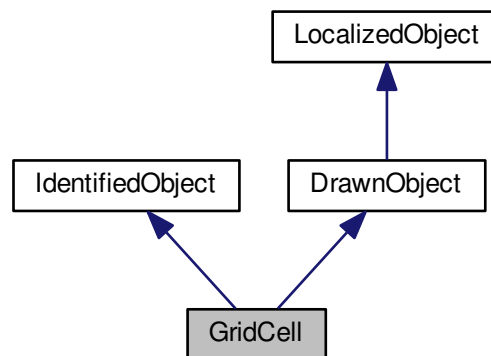
- [src/socialSpace/GaussianSpace.h](#)
- [src/socialSpace/GaussianSpace.cpp](#)

5.7 GridCell Class Reference

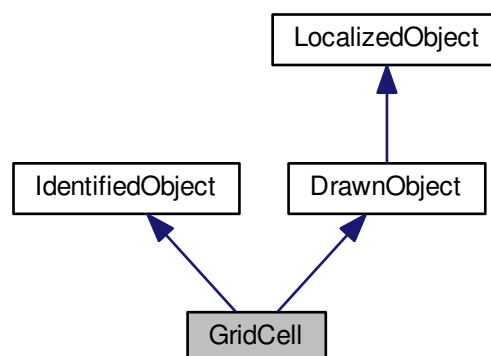
This class represent a cell in the [GridMap](#).

```
#include <GridCell.h>
```

Inheritance diagram for GridCell:



Collaboration diagram for GridCell:



Public Member Functions

- [GridCell](#) (double size, double value=0, Vector3d [position](#)=Vector3d(), int [id](#)=0)

Constructor.

- virtual [~GridCell](#) ()

Destructor.

- double [getValue](#) () const

Simple getter.

- void [setValue](#) (double value)

Simple setter.

- double [getSize](#) () const

Simple getter.

- void [setSize](#) (double size)

Simple setter.

- bool [isBorderEnabled](#) () const

Simple getter.

- void [setBorderEnabled](#) (bool [borderEnabled](#)=true)

Simple setter.

- double [getAStarScore](#) () const

Simple getter.

- void [setAStarScore](#) (int starScore)

Simple setter.

- bool [isCellSelected](#) () const

Simple getter.

- void [setCellSelected](#) (bool [cellSelected](#)=true)

Simple setter.

- bool [isFrontier](#) () const

Simple getter.

- void [setFrontier](#) (bool [frontier](#)=true)

Simple setter.

- bool [isProcessed](#) () const

Simple getter.

- void [setProcessed](#) (bool [processed](#)=true)

Simple setter.

- bool [isGoal](#) () const

Simple getter.

- void [setGoal](#) (bool [goal](#)=true)

Simple setter.

- bool [isStart](#) () const

Simple getter.

- void [setStart](#) (bool [start](#)=true)

Simple setter.

- bool [isInfoEnabled](#) () const

Simple getter.

- void [setInfoEnabled](#) (bool [infoEnabled](#)=true)

Simple setter.

Protected Attributes

- double `_size`
The size of the `GridCell`.
- double `_value`
The value of the `GridCell`.
- bool `borderEnabled` = false
Enable of disable `GridCell` border drawing.
- double `_aStarScore` = -1
The A* score of the `GridCell`.
- bool `cellSelected` = false
Select of deselect `GridCell` (change drawing color)
- bool `frontier` = false
If the `GridCell` is a frontier (change drawing color)
- bool `processed` = false
If the `GridCell` has been processed (change drawing color)
- bool `goal` = false
If the `GridCell` is the goal to reach (change drawing color)
- bool `start` = false
If the `GridCell` is the starting point (change drawing color)
- bool `infoEnabled` = false
Enable of disable `GridCell` informations drawing.

5.7.1 Detailed Description

This class represent a cell in the `GridMap`.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 `GridCell::GridCell (double size, double value = 0, Vector3d position = Vector3d(), int id = 0)`

Constructor.

Constructor of the `GridCell` class

Parameters

<code>size</code>	: The size of the <code>GridCell</code>
<code>value</code>	: The initial value of the <code>GridCell</code>
<code>position</code>	: The position of the <code>GridCell</code> in real coordinate
<code>id</code>	: The unique identifier of the <code>GridCell</code>

5.7.2.2 `GridCell::~GridCell () [virtual]`

Destructor.

Destructor of the `GridCell` class

5.7.3 Member Function Documentation

5.7.3.1 double GridCell::getAScore () const

Simple getter.

Returns

aStarScore

5.7.3.2 double GridCell::getSize () const

Simple getter.

Returns

size

5.7.3.3 double GridCell::getValue () const

Simple getter.

Returns

value

5.7.3.4 bool GridCell::isBorderEnabled () const

Simple getter.

Returns

borderEnabled

5.7.3.5 bool GridCell::isCellSelected () const

Simple getter.

Returns

cellSelected

5.7.3.6 bool GridCell::isFrontier () const

Simple getter.

Returns

frontier

5.7.3.7 bool GridCell::isGoal () const

Simple getter.

Returns

goal

5.7.3.8 bool GridCell::isInfoEnabled () const

Simple getter.

Returns

infoEnabled

5.7.3.9 bool GridCell::isProcessed () const

Simple getter.

Returns

processed

5.7.3.10 bool GridCell::isStart () const

Simple getter.

Returns

start

5.7.3.11 void GridCell::setAScore (int *starScore*)

Simple setter.

Parameters

<i>starScore</i>	
------------------	--

5.7.3.12 void GridCell::setBorderEnabled (bool *borderEnabled* = true)

Simple setter.

Parameters

<i>borderEnabled</i>	
----------------------	--

5.7.3.13 void GridCell::setCellSelected (bool *cellSelected* = true)

Simple setter.

Parameters

<i>cellSelected</i>	
---------------------	--

5.7.3.14 void GridCell::setFrontier (bool *frontier* = true)

Simple setter.

Parameters

<i>frontier</i>	
-----------------	--

5.7.3.15 void GridCell::setGoal (bool *goal* = true)

Simple setter.

Parameters

<i>goal</i>	
-------------	--

5.7.3.16 void GridCell::setInfoEnabled (bool *infoEnabled* = true)

Simple setter.

Parameters

<i>infoEnabled</i>	
--------------------	--

5.7.3.17 void GridCell::setProcessed (bool *processed* = true)

Simple setter.

Parameters

<i>processed</i>	
------------------	--

5.7.3.18 void GridCell::setSize (double *size*)

Simple setter.

Parameters

<i>size</i>	
-------------	--

5.7.3.19 void GridCell::setStart (bool *start* = true)

Simple setter.

Parameters

<i>start</i>	
--------------	--

5.7.3.20 void GridCell::setValue (double *value*)

Simple setter.

Parameters

<i>value</i>	
--------------	--

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

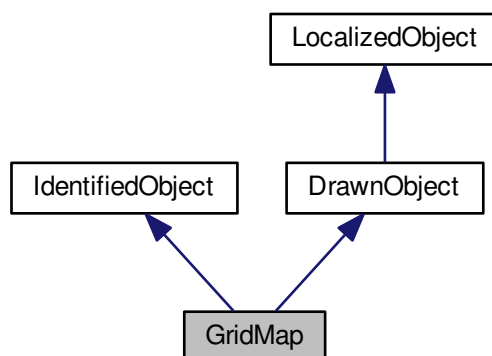
- [src/worldRepresentation/GridCell.h](#)
- [src/worldRepresentation/GridCell.cpp](#)

5.8 GridMap Class Reference

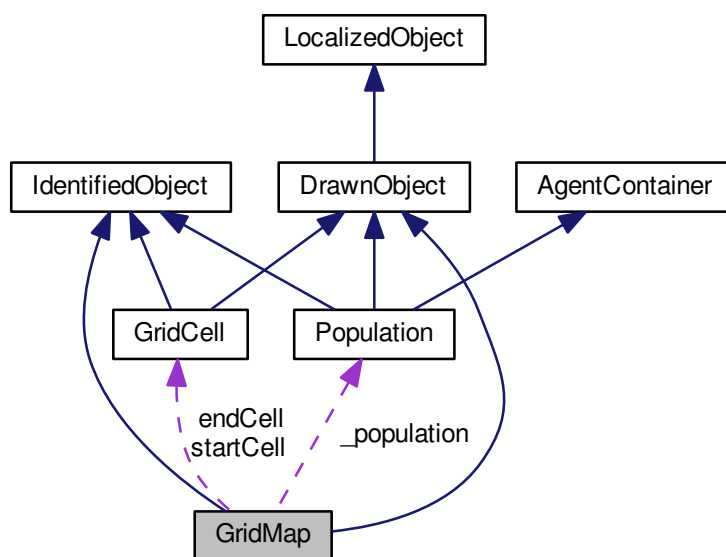
This class manage the 2D [GridMap](#) computed from the Agents [SocialSpace](#).

```
#include <GridMap.h>
```

Inheritance diagram for GridMap:



Collaboration diagram for GridMap:



Classes

- struct [CompaireVCell](#)

Define an operator for [GridCell](#) and associated gScore comparison for A* algorithm.

Public Member Functions

- [GridMap](#) ([World](#) *world, [Population](#) *pop, double [resolution](#)=0.1)
Constructor.
- virtual [~GridMap](#) ()
Destructor.
- void [compute](#) ()
Compute the value of each [GridCell](#) in the [GridMap](#) based on Agents [SocialSpace](#).
- void [normalize](#) ()
Normalize the [GridCells](#) value between 0 and 1.
- void [update](#) ()
Update the [GridMap](#) by calling compute and normalize function.
- void [deselectCells](#) ()
Set all [GridCell](#) to selected False.
- int [pathFinderNextStep](#) ()
Compute one step of the A algorithm.*
- void [resetCellColor](#) ()
Reset the color of all [GridCell](#).
- std::vector< [GridCell](#) * > [constructPath](#) ()
Build the path from A algorithm result.*
- std::vector< [GridCell](#) * > [neighbors](#) ([GridCell](#) *cell, bool allowDiagonalMove=true)
Find the neighbors of a [GridCell](#) in the [GridMap](#).
- std::vector< [GridCell](#) * > [findPath](#) ([GridCell](#) *startCell, [GridCell](#) *endCell)
Find path between two given [GridCell](#) in parameters.
- [GridCell](#) * [getCell](#) (unsigned int cellId)
Get a [GridCell](#) in the [GridMap](#) by id.
- [GridCell](#) * [getCell](#) (double x, double y)
find a [GridCell](#) in the [GridMap](#) from its real coordinates
- void [setInfoEnabled](#) (bool [infoEnabled](#)=false)
Simple setter.
- bool [isGroupSpaceEnabled](#) () const
Simple getter.
- void [setGroupSpaceEnabled](#) (bool [groupSpaceEnabled](#)=true)
Simple setter.
- bool [isPersonalSpaceEnabled](#) () const
Simple getter.
- void [setPersonalSpaceEnabled](#) (bool [personalSpaceEnabled](#)=true)
Simple setter.
- bool [isBorderEnabled](#) () const
Simple getter.
- void [setBorderEnabled](#) (bool [borderEnabled](#)=true)
Simple setter.

Protected Types

- typedef std::pair< double, [GridCell](#) * > [VCell](#)
Define an operator for [GridCell](#) and associated gScore comparison for A algorithm.*

Protected Attributes

- **Population** * **_population**
The *Population* related to the *GridMap*.
- double **width**
The real width of the *GridMap* in meter.
- double **height**
The real height of the *GridMap* in meter.
- double **resolution**
The resolution of the *GridMap*.
- double **minValue**
The minimum value of a *GridCell* in the *GridMap*.
- double **maxValue**
The maximum value of a *GridCell* in the *GridMap*.
- bool **personalSpaceEnabled** = true
Enable or disable the computing of the *PersonalSocialSpace* in the *GridMap*.
- bool **groupSpaceEnabled** = true
Enable or disable the computing of the *GroupSocialSpace* in the *GridMap*.
- bool **borderEnabled** = false
Enable or disable the drawing of the *GridCell* border.
- bool **infoEnabled** = false
Enable or disable the drawing of the *GridCell* information.
- std::priority_queue< **VCell**, std::vector< **VCell** >, **CompaireVCell** > **openNodesPQ**
Priority queue for open nodes processing in A* algorithm.
- std::map< **GridCell** *, **GridCell** * > **cameFrom**
Path relation between *GridCell* for A* algorithm.
- std::map< **GridCell** *, double > **gScore**
The score obtained in A* algorithm associated to its *GridCell*.
- **GridCell** * **endCell**
The goal *GridCell* for the path.
- **GridCell** * **startCell**
The starting *GridCell* for the path.

5.8.1 Detailed Description

This class manage the 2D *GridMap* computed from the Agents *SocialSpace*.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 **GridMap::GridMap** (**World** * *world*, **Population** * *pop*, double *resolution* = 0.1)

Constructor.

Constructor of the *GridMap* class

Parameters

<i>world</i>	: The main frame coordinates
<i>pop</i>	: The population related to this <i>GridMap</i>
<i>resolution</i>	: The resolution of the <i>GridMap</i> , must be a fraction of the <i>World</i> width and height

5.8.2.2 GridMap::~GridMap () [virtual]

Destructor.

Destructor of the [GridMap](#) class

5.8.3 Member Function Documentation

5.8.3.1 std::vector< GridCell * > GridMap::constructPath ()

Build the path from A* algorithm result.

Returns

List of [GridCell](#) representing the path found in the [GridMap](#)

5.8.3.2 std::vector< GridCell * > GridMap::findPath (GridCell * startCell, GridCell * endCell)

Find path between two given [GridCell](#) in parameters.

Parameters

<i>startCell</i>	: The starting GridCell
<i>endCell</i>	: The goal GridCell

Returns

The path found in the [GridMap](#) in a list of [GridCell](#)

5.8.3.3 GridCell * GridMap::getCell (unsigned int cellId)

Get a [GridCell](#) in the [GridMap](#) by id.

Parameters

<i>cellId</i>	: The unique identifier of the GridCell
---------------	---

Returns

The corresponding [GridCell](#), null if not found

5.8.3.4 GridCell * GridMap::getCell (double x, double y)

find a [GridCell](#) in the [GridMap](#) from its real coordinates

Parameters

<i>x</i>	: The x coordinate of the GridCell in meter
<i>y</i>	: The y coordinate of the GridCell in meter

Returns

The corresponding [GridCell](#), null if not found

5.8.3.5 `bool GridMap::isBorderEnabled () const`

Simple getter.

Returns

`borderEnabled`

5.8.3.6 `bool GridMap::isGroupSpaceEnabled () const`

Simple getter.

Returns

`groupSpaceEnabled`

5.8.3.7 `bool GridMap::isPersonalSpaceEnabled () const`

Simple getter.

Returns

`personalSpaceEnabled`

5.8.3.8 `std::vector< GridCell * > GridMap::neighbors (GridCell * cell, bool allowDiagonalMove = true)`

Find the neighbors of a [GridCell](#) in the [GridMap](#).

Parameters

<i>cell</i>	: Target GridCell
<i>allowDiagonalMove</i>	: Allow diagonal neighboring

Returns

List of neighbors [GridCell](#)

5.8.3.9 int GridMap::pathFinderNextStep ()

Compute one step of the A* algorithm.

Returns

1 if pathFinder reached goal, -1 pathFinder couldn't reach the goal, 0 otherwise

5.8.3.10 void GridMap::setBorderEnabled (bool *borderEnabled* = true)

Simple setter.

Parameters

<i>borderEnabled</i>	
----------------------	--

5.8.3.11 void GridMap::setGroupSpaceEnabled (bool *groupSpaceEnabled* = true)

Simple setter.

Parameters

<i>groupSpaceEnabled</i>	
--------------------------	--

5.8.3.12 void GridMap::setInfoEnabled (bool *infoEnabled* = false)

Simple setter.

Parameters

<i>infoEnabled</i>	
--------------------	--

5.8.3.13 void GridMap::setPersonalSpaceEnabled (bool *personalSpaceEnabled* = true)

Simple setter.

Parameters

<i>personalSpaceEnabled</i>	
-----------------------------	--

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

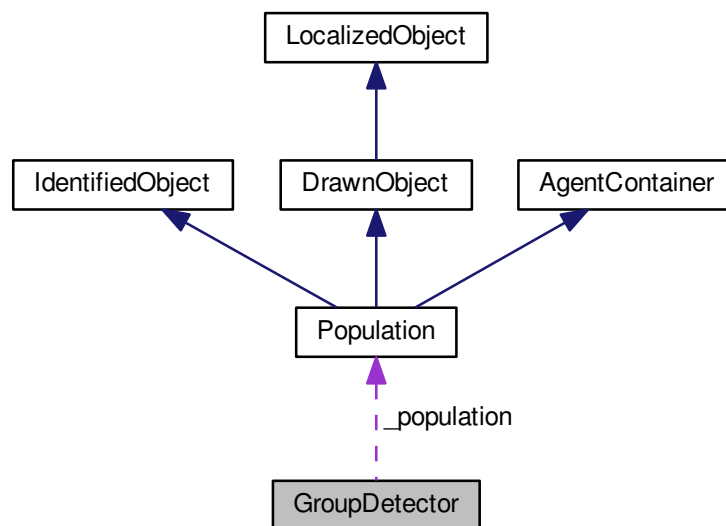
- [src/worldRepresentation/GridMap.h](#)
- [src/worldRepresentation/GridMap.cpp](#)

5.9 GroupDetector Class Reference

This class is dedicated to process every Agents in the [Population](#) and create every [Formation](#).

```
#include <GroupDetector.h>
```

Collaboration diagram for GroupDetector:



Public Member Functions

- [GroupDetector](#) ([Population](#) *pop)
Constructor.
- virtual [~GroupDetector](#) ()
Destructor.
- void [detect](#) ()
Find and create every Formations in the [Population](#).
- void [checkExistingFormation](#) ()
Check if existing Formations are empty or if [Agent](#) is alone in a [Formation](#) and remove them.

Protected Attributes

- [Population](#) * [_population](#)
The [Population](#) processed by the [GroupDetector](#).
- int [formationId](#) = 0
The actual increment for unique identifier of the Formations created.

5.9.1 Detailed Description

This class is dedicated to process every Agents in the [Population](#) and create every [Formation](#).

Todo This class should be an interface and actual logic should be implemented has BasicGroupDetector

5.9.2 Constructor & Destructor Documentation

5.9.2.1 GroupDetector::GroupDetector ([Population](#) * *pop*)

Constructor.

Constructor of the [GroupDetector](#) class

Parameters

<i>pop</i>	: The Population that the detector will process
------------	---

5.9.2.2 GroupDetector::~~GroupDetector () [virtual]

Destructor.

Destructor of the [GroupDetector](#) class, it does not call the related [Population](#) destructor

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

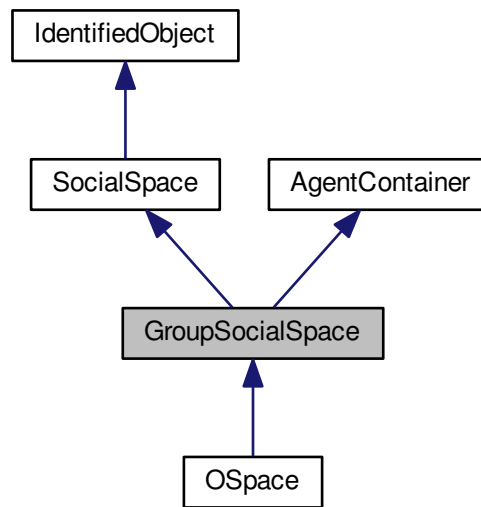
- [src/socialProcessing/GroupDetector.h](#)
- [src/socialProcessing/GroupDetector.cpp](#)

5.10 GroupSocialSpace Class Reference

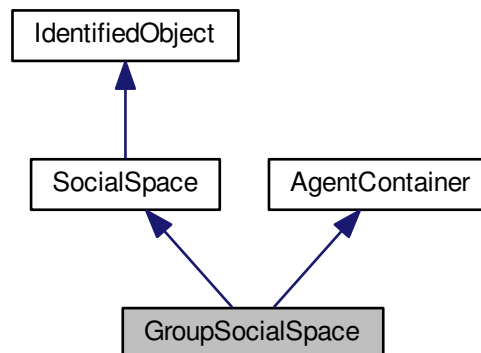
This class is an interface to implement representation of a [GroupSocialSpace](#).

```
#include <GroupSocialSpace.h>
```

Inheritance diagram for GroupSocialSpace:



Collaboration diagram for GroupSocialSpace:



Public Member Functions

- [GroupSocialSpace](#) (int id=0)
Constructor.
- [GroupSocialSpace](#) (std::vector< [Agent](#) * > &agents, int id=0)
Constructor.
- virtual [~GroupSocialSpace](#) ()
Destructor.

Additional Inherited Members

5.10.1 Detailed Description

This class is an interface to implement representation of a [GroupSocialSpace](#).

This class is an interface to implement representation of the [GroupSocialSpace](#) of a [Formation](#)

5.10.2 Constructor & Destructor Documentation

5.10.2.1 `GroupSocialSpace::GroupSocialSpace (int id = 0)`

Constructor.

Constructor of the [GroupSocialSpace](#) class, with no [Agent](#)

Parameters

<i>id</i>	: The unique identifier of the GroupSocialSpace
-----------	---

5.10.2.2 `GroupSocialSpace::GroupSocialSpace (std::vector< Agent * > & agents, int id = 0)`

Constructor.

Constructor of the [GroupSocialSpace](#) class, for the Agents in parameter

Parameters

<i>agents</i>	: The Agents related to the GroupSocialSpace
<i>id</i>	: The unique identifier of the GroupSocialSpace

5.10.2.3 `GroupSocialSpace::~~GroupSocialSpace ()` [virtual]

Destructor.

Destructor of the [GroupSocialSpace](#) class

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

- `src/socialSpace/GroupSocialSpace.h`
- `src/socialSpace/GroupSocialSpace.cpp`

5.11 Gui Class Reference

Graphical User Interface.

```
#include <Gui.h>
```

Public Member Functions

- void **draw** ()
- void **buttonPressed** ()
- void **togglePressed** (bool &pressed)
- void **exit** ()

Public Attributes

- ofxButton **button**
- ofxToggle **toggle**
- ofxPanel **gui**

5.11.1 Detailed Description

Graphical User Interface.

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

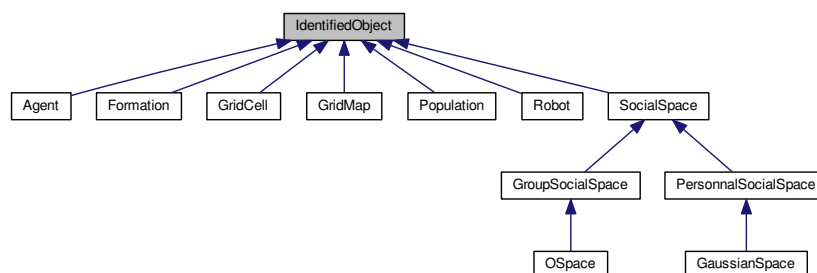
- src/Gui.h
- src/Gui.cpp

5.12 IdentifiedObject Class Reference

This class is an interface for class that need a unique identifier.

```
#include <IdentifiedObject.h>
```

Inheritance diagram for IdentifiedObject:



Public Member Functions

- **IdentifiedObject** (unsigned int **id**=0)
Constructor.
- virtual **~IdentifiedObject** ()
Destructor.
- unsigned int **getId** () const
Simple getter.
- void **setId** (unsigned int **id**)
Simpler setter.

Protected Attributes

- unsigned int [id](#)

The unique identifier.

5.12.1 Detailed Description

This class is an interface for class that need a unique identifier.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 IdentifiedObject::IdentifiedObject (unsigned int *id* = 0)

Constructor.

Constructor of the [IdentifiedObject](#) class

Parameters

<i>id</i>	: The unique identifier of the object
-----------	---------------------------------------

5.12.2.2 IdentifiedObject::~~IdentifiedObject () [virtual]

Destructor.

Destructor of the [IdentifiedObject](#) class

5.12.3 Member Function Documentation

5.12.3.1 unsigned int IdentifiedObject::getId () const

Simple getter.

Returns

id

5.12.3.2 void IdentifiedObject::setId (unsigned int *id*)

Simpler setter.

Parameters

<i>id</i>	
-----------	--

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

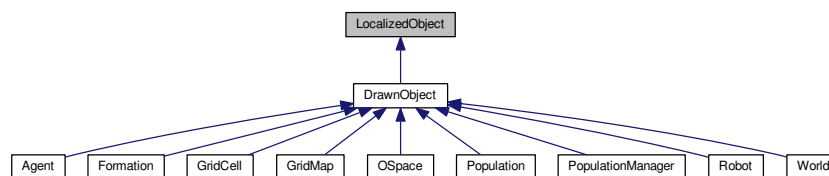
- [src/genericType/IdentifiedObject.h](#)
- [src/genericType/IdentifiedObject.cpp](#)

5.13 LocalizedObject Class Reference

This class is an interface for object that are localized in real [World](#).

```
#include <LocalizedObject.h>
```

Inheritance diagram for LocalizedObject:



Public Member Functions

- [LocalizedObject](#) (Vector3d [position](#)=Vector3d(), double [theta](#)=0)
Constructor.
- virtual [~LocalizedObject](#) ()
Destructor.
- Vector3d [getDirection](#) ()
Compute the direction vector of the object from its angle.
- double [getX](#) () const
Simple getter.
- void [setX](#) (double x)
Simple setter.
- double [getY](#) () const
Simple getter.
- void [setY](#) (double y)
Simple setter.
- Vector3d [getPosition](#) () const
Simple getter.
- void [setPosition](#) (Vector3d [position](#))
Simple setter.
- double [getTheta](#) () const
Simple getter.
- void [setTheta](#) (double [theta](#))
Simple setter.

Protected Attributes

- Vector3d [position](#)
The position of the object.
- double [theta](#)
The angle of the object relative to X axis (1,0,0) (arround Z axis (0,0,1)) and in radian normalized between [0,2PI].

5.13.1 Detailed Description

This class is an interface for object that are localized in real [World](#).

This class is an interface for object that are localize in real [World](#) with x, y, z and Theta coordinates

5.13.2 Constructor & Destructor Documentation

5.13.2.1 LocalizedObject::LocalizedObject (Vector3d *position* = Vector3d(), double *theta* = 0)

Constructor.

Constructor of the [LocalizedObject](#) class

Parameters

<i>position</i>	: The initial position of the object
<i>theta</i>	: The initial angle of the object relative to X axis in radian

5.13.2.2 LocalizedObject::~~LocalizedObject () [virtual]

Destructor.

Destructor of the [LocalizedObject](#) class

5.13.3 Member Function Documentation

5.13.3.1 Vector3d LocalizedObject::getDirection ()

Compute the direction vector of the object from its angle.

Returns

The direction Vector3d

5.13.3.2 Vector3d LocalizedObject::getPosition () const

Simple getter.

Returns

position

5.13.3.3 double LocalizedObject::getTheta () const

Simple getter.

Returns

theta

5.13.3.4 double LocalizedObject::getX () const

Simple getter.

Returns

X coordinate of the position vector

5.13.3.5 double LocalizedObject::getY () const

Simple getter.

Returns

Y coordinate of the position vector

5.13.3.6 void LocalizedObject::setPosition (Vector3d *position*)

Simple setter.

Parameters

<i>position</i>	
-----------------	--

5.13.3.7 void LocalizedObject::setTheta (double *theta*)

Simple setter.

Parameters

<i>theta</i>	
--------------	--

5.13.3.8 void LocalizedObject::setX (double x)

Simple setter.

Parameters

<i>x</i>	
----------	--

5.13.3.9 void LocalizedObject::setY (double y)

Simple setter.

Parameters

<i>y</i>	
----------	--

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

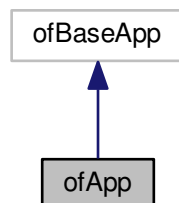
- [src/genericType/LocalizedObject.h](#)
- [src/genericType/LocalizedObject.cpp](#)

5.14 ofApp Class Reference

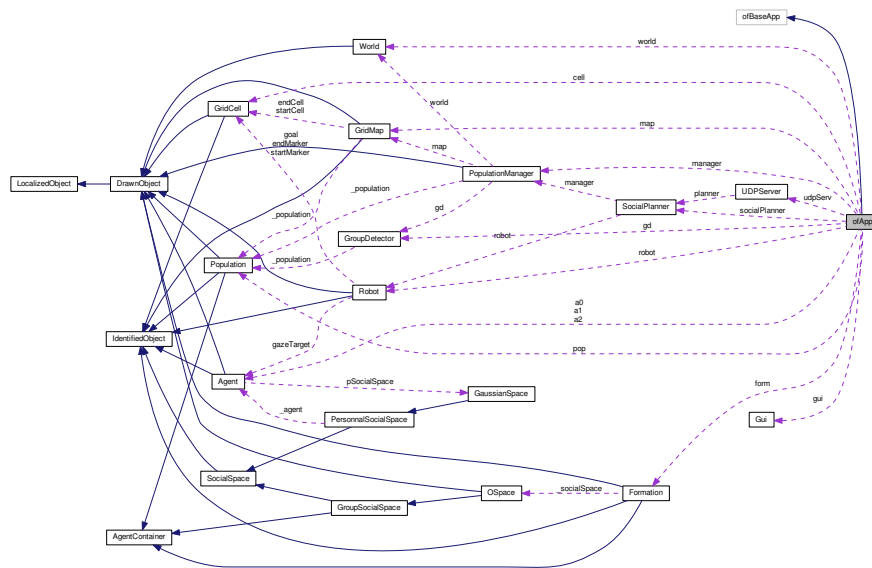
The Openframeworks main class.

```
#include <ofApp.h>
```

Inheritance diagram for ofApp:



Collaboration diagram for ofApp:



Public Member Functions

- void **setup** ()
- void **update** ()
- void **draw** ()
- void **keyPressed** (int key)
- void **keyReleased** (int key)
- void **mouseMoved** (int x, int y)
- void **mouseDragged** (int x, int y, int button)
- void **mousePressed** (int x, int y, int button)
- void **mouseReleased** (int x, int y, int button)
- void **mouseEntered** (int x, int y)
- void **mouseExited** (int x, int y)
- void **windowResized** (int w, int h)
- void **dragEvent** (ofDragInfo dragInfo)
- void **gotMessage** (ofMessage msg)
- void **exit** ()
- void **buttonPressed** ()

Public Attributes

- **Gui** * **gui**
- std::thread **server_thread**
- **UDPServer** * **udpServ**
- **Robot** * **robot**
- **SocialPlanner** * **socialPlanner**
- **Population** * **pop**
- **PopulationManager** * **manager**
- **Formation** * **form**
- std::vector< **Agent** * > **agents**

- [Agent](#) * **a0**
- [Agent](#) * **a1**
- [Agent](#) * **a2**
- [World](#) * **world**
- [GridMap](#) * **map**
- [GridCell](#) * **cell**
- [GroupDetector](#) * **gd**
- unsigned int **mainIndex** = 0

5.14.1 Detailed Description

The Openframeworks main class.

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

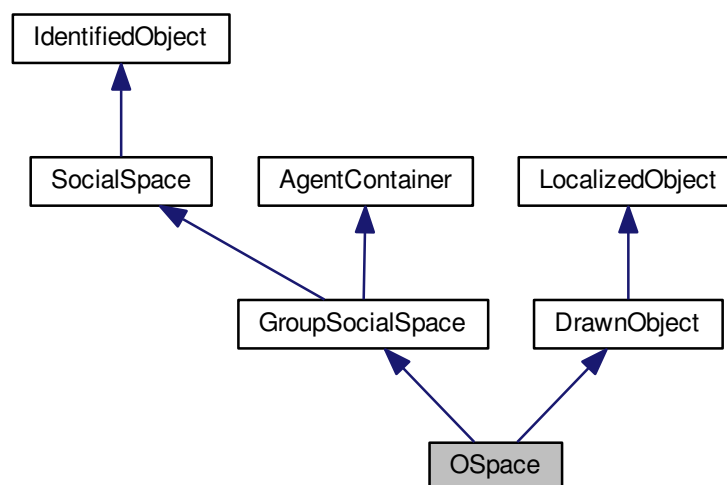
- src/ofApp.h
- src/ofApp.cpp

5.15 OSpace Class Reference

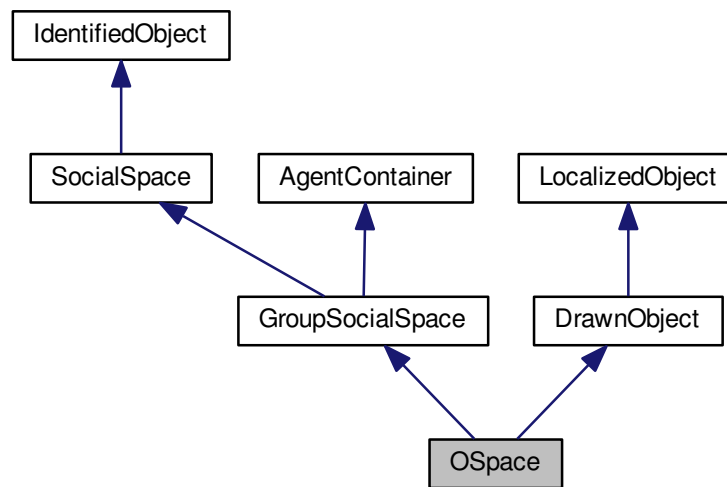
This class is an implementation of the [GroupSocialSpace](#).

```
#include <OSpace.h>
```

Inheritance diagram for OSpace:



Collaboration diagram for OSpace:



Public Member Functions

- `OSpace (int id=0)`
Constructor.
- `OSpace (std::vector< Agent * > &agents, int id=0)`
Constructor.
- `~OSpace ()`
Destructor.
- `void computeCenter ()`
Compute the gravity center of the OSpace based on Agents positions.
- `void sortAgents ()`
Sort AgentContainer of the OSpace in clockwise order relative to gravity center.
- `void computeCentroids ()`
Compute the centroids of each pair of Agent of the OSpace based on the field of view of every Agent.
- `void computeCenter ()`
Compute the center of the Formation based on centroids.
- `void computeCovarMatrix ()`
Compute the covariance matrix of the 2D gaussian mixture model.
- `void update ()`
Update the OSpace by computing its gravity center, centroids and deduce the covariance matrix of the 2D gaussian mixture model.
- `double phi (Vector3d testedPoint)`
Compute the value of the OSpace at a given point in space.
- `bool less (Vector3d a, Vector3d b)`
Compare two vector based on the OSpace gravity center.
- `Vector3d getCenter () const`
Simple getter.
- `void setCenter (const Vector3d ¢er)`

Simpler setter.

- Vector3d [getgCenter](#) () const

Simple getter.

- void [setgCenter](#) (const Vector3d &[gCenter](#))

Simple setter.

Protected Attributes

- std::vector< std::vector< Vector3d > > [dh_seg](#)

List of DH vector representing the interpersonnal distance between the Agents.

- std::vector< std::vector< Vector3d > > [di_seg](#)

List of DI vector representing the distance between field of view intersection of Agents.

- Vector3d [center](#)

The center of the OSpace.

- Vector3d [gCenter](#)

The gravity center based on the Agents position.

- double [rotation](#) = 0.0f

The rotation applied to the 2D gaussian mixture model.

- std::vector< Vector3d > [intersectionPoints](#)

List of field of view intersection point between Agents.

- std::vector< Vector3d > [centroids](#)

List of the centroid points based on intersection points and Agents position.

- Matrix< double, 2, 2 > [covarMatrix](#)

The covariance matrix of the 2D gaussian mixture model.

5.15.1 Detailed Description

This class is an implementation of the [GroupSocialSpace](#).

This class is an implementation of the [GroupSocialSpace](#) represented by a 2D gaussian mixture model based on interpersonal distances and a algorithm to find the center of the related [Formation](#).

5.15.2 Constructor & Destructor Documentation

5.15.2.1 OSpace::OSpace (int *id* = 0)

Constructor.

Constructor of the [OSpace](#) class, with no [Agent](#)

Parameters

<i>id</i>	: The unique identifier of the OSpace
-----------	---

5.15.2.2 OSpace::OSpace (std::vector< Agent * > &*agents*, int *id* = 0)

Constructor.

Constructor of the [OSpace](#) class, for the Agents in parameter

Parameters

<i>agents</i>	: The Agents related to the GroupSocialSpace
<i>id</i>	: The unique identifier of the GroupSocialSpace

5.15.2.3 OSpace::~~OSpace ()

Destructor.

Destructor of the [OSpace](#) class

5.15.3 Member Function Documentation

5.15.3.1 void OSpace::computeCovarMatrix ()

Compute the covariance matrix of the 2D gaussian mixture model.

Compute the covariance matrix of the 2D gaussian mixture model based on interpersonnal distance between Agents

5.15.3.2 Vector3d OSpace::getCenter () const

Simple getter.

Returns

center

5.15.3.3 Vector3d OSpace::getgCenter () const

Simple getter.

Returns

gCenter

5.15.3.4 bool OSpace::less (Vector3d *a*, Vector3d *b*)

Compare two vector based on the [OSpace](#) gravity center.

Compare two vector based on the [OSpace](#) gravity center to order them clockwise

Parameters

<i>a</i>	: The first Agent position
<i>b</i>	: The second Agent position

Returns

1 if a [Agent](#) is before b [Agent](#) in clockwise order relative to gCenter, 0 otherwise

5.15.3.5 double OSpace::phi (Vector3d *testedPoint*)

Compute the value of the [OSpace](#) at a given point in space.

Parameters

<i>testedPoint</i>	: The coordinates of the point in the real frame coordinates World
--------------------	--

Returns

The value of the [OSpace](#) at the given point

5.15.3.6 void OSpace::setCenter (const Vector3d & *center*)

Simpler setter.

Parameters

<i>center</i>	
---------------	--

5.15.3.7 void OSpace::setgCenter (const Vector3d & *gCenter*)

Simple setter.

Parameters

<i>gCenter</i>	
----------------	--

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

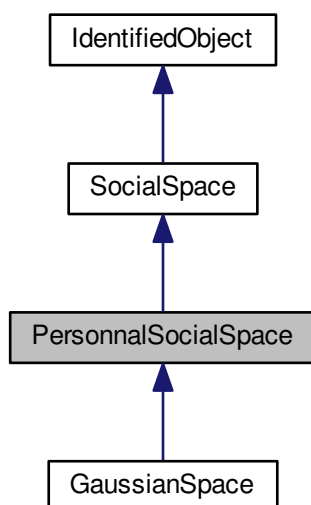
- [src/socialSpace/OSpace.h](#)
- [src/socialSpace/OSpace.cpp](#)

5.16 PersonnalSocialSpace Class Reference

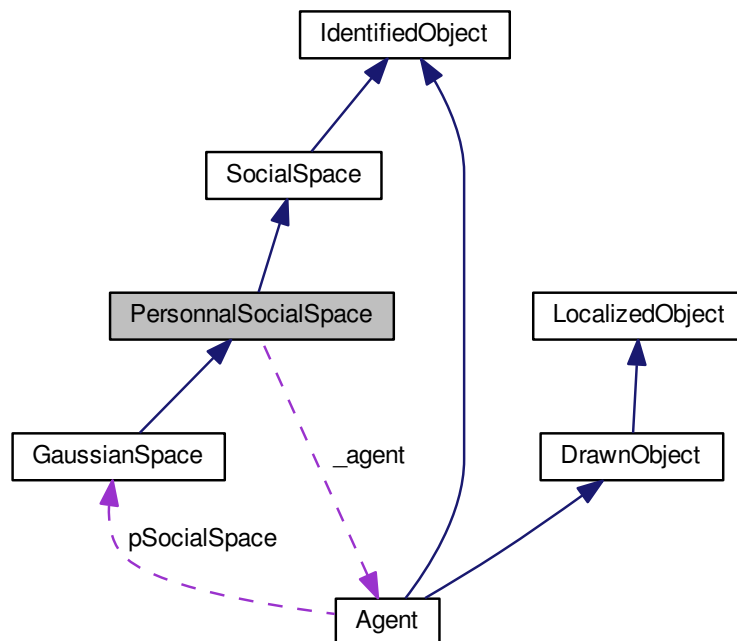
This class is an interface to implement representation of a [PersonnalSocialSpace](#).

```
#include <PersonnalSocialSpace.h>
```

Inheritance diagram for PersonnalSocialSpace:



Collaboration diagram for PersonnalSocialSpace:



Public Member Functions

- [PersonnalSocialSpace](#) ([Agent](#) *agent, int id=0)
Constructor.
- virtual [~PersonnalSocialSpace](#) ()
Destructor.

Protected Attributes

- [Agent](#) * [_agent](#)
The [Agent](#) related to the [PersonnalSocialSpace](#).

5.16.1 Detailed Description

This class is an interface to implement representation of a [PersonnalSocialSpace](#).

This class is an interface to implement representation of the [PersonnalSocialSpace](#) of an [Agent](#)

5.16.2 Constructor & Destructor Documentation

5.16.2.1 [PersonnalSocialSpace::PersonnalSocialSpace](#) ([Agent](#) * agent, int id = 0)

Constructor.

Constructor of the [PersonnalSocialSpace](#) class

Parameters

<i>agent</i>	: The Agent related to the PersonnalSocialSpace
<i>id</i>	: The unique identifier of the SocialSpace

5.16.2.2 `PersonnalSocialSpace::~~PersonnalSocialSpace () [virtual]`

Destructor.

Destructor of the [PersonnalSocialSpace](#) class

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

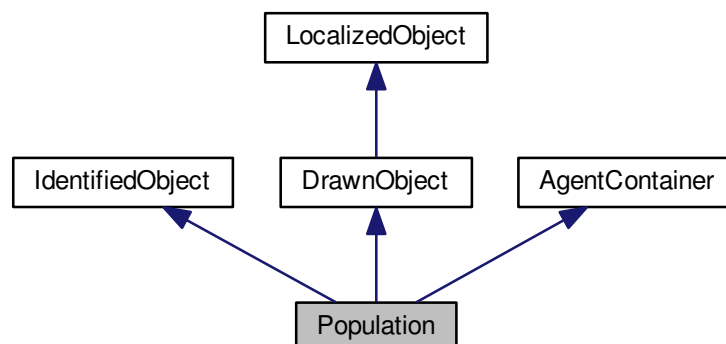
- `src/socialSpace/PersonnalSocialSpace.h`
- `src/socialSpace/PersonnalSocialSpace.cpp`

5.17 Population Class Reference

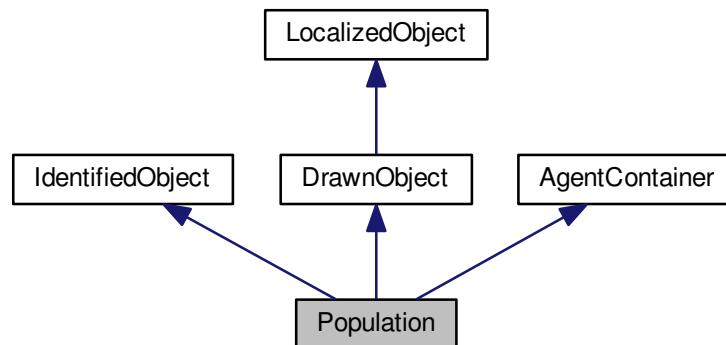
This class represent [Population](#) around the [Robot](#).

```
#include <Population.h>
```

Inheritance diagram for Population:



Collaboration diagram for Population:



Public Member Functions

- **Population** (std::vector< **Agent** * > agents, Vector3d **position**=Vector3d(), int **id**=0)
Constructor.
- **Population** (Vector3d **position**=Vector3d(), int **id**=0)
Constructor.
- **~Population** ()
Destructor.
- **Formation** * **getHighestFormationInteractionPotential** ()
*Find the **Formation** with the highest interaction potential.*
- **Formation** * **getRelatedFormation** (unsigned int agentId)
*Find the **Formation** related to the **Agent**.*
- **Formation** * **getRelatedFormation** (**Agent** *a)
*Find the **Formation** related to the **Agent** Alias to getRelatedFormation function by **Agent** id.*
- void **clear** ()
*Clear the **Population**.*
- int **isGrouped** (**Agent** *agent)
*Check if the **Agent** is part of a **Formation** Alias to isGrouped function by **Agent** id.*
- int **isGrouped** (unsigned int agentId)
*Check if the **Agent** is part of a **Formation**.*
- void **pushFormation** (**Formation** *formation)
*Add a **Formation** to the **Population**.*
- int **removeFormation** (unsigned int formationId)
*Remove a **Formation** by id from the **Population** and the corresponding **GroupSocialSpace**.*
- void **clearFormations** ()
*Clear the **Formation** list.*
- const std::vector< **Formation** * > & **getFormations** () const
Simple getter.
- void **setFormations** (const std::vector< **Formation** * > &formations)
Simpler setter.

Additional Inherited Members

5.17.1 Detailed Description

This class represent [Population](#) around the [Robot](#).

This class represent [Population](#) around the [Robot](#), it contains all the Agents with the Formations detected in this [Population](#).

5.17.2 Constructor & Destructor Documentation

5.17.2.1 `Population::Population (std::vector< Agent * > agents, Vector3d position = Vector3d(), int id = 0)`

Constructor.

Constructor of the [Population](#) class

Parameters

<i>agents</i>	: List of initial Agents that are part of the Population
<i>position</i>	: Position of the Population (required by DrawnObject but useless here)
<i>id</i>	: Unique identifier of the Population

5.17.2.2 `Population::Population (Vector3d position = Vector3d(), int id = 0)`

Constructor.

Constructor of the [Population](#) class

Parameters

<i>position</i>	: Position of the Population (required by DrawnObject but useless here)
<i>id</i>	: Unique identifier of the Population

5.17.2.3 `Population::~~Population ()`

Destructor.

Destructor of the [Population](#) class, destroy every Agents and Formations related to this [Population](#)

5.17.3 Member Function Documentation

5.17.3.1 `void Population::clear ()`

Clear the [Population](#).

Destroy all Agents and Formations

5.17.3.2 void Population::clearFormations ()

Clear the [Formation](#) list.

This function do not call the [Formation](#) destructor

5.17.3.3 const std::vector< Formation * > & Population::getFormations () const

Simple getter.

Returns

Formations

5.17.3.4 Formation* Population::getHighestFormationInteractionPotential ()

Find the [Formation](#) with the highest interaction potential.

Returns

The [Formation](#) with the highest interaction potential in the [Population](#)

5.17.3.5 Formation * Population::getRelatedFormation (unsigned int *agentId*)

Find the [Formation](#) related to the [Agent](#).

Returns

The [Formation](#) in which the [Agent](#) is taking part

5.17.3.6 Formation * Population::getRelatedFormation (Agent * *a*)

Find the [Formation](#) related to the [Agent](#) Alias to getRelatedFormation function by [Agent](#) id.

Returns

The [Formation](#) in which the [Agent](#) is taking part

5.17.3.7 int Population::isGrouped (Agent * *agent*)

Check if the [Agent](#) is part of a [Formation](#) Alias to isGrouped function by [Agent](#) id.

Parameters

<i>agent</i>	: The Agent to check
--------------	--------------------------------------

Returns

1 if [Agent](#) is in a [Formation](#), 0 otherwise

5.17.3.8 int Population::isGrouped (unsigned int *agentId*)

Check if the [Agent](#) is part of a [Formation](#).

Parameters

<i>agent↔ Id</i>	: The Agent to check
----------------------	--------------------------------------

Returns

1 if [Agent](#) is in a [Formation](#), 0 otherwise

5.17.3.9 void Population::pushFormation ([Formation](#) * *formation*)

Add a [Formation](#) to the [Population](#).

Parameters

<i>formation</i>	: The Formation to add to the Population
------------------	--

5.17.3.10 int Population::removeFormation (unsigned int *formationId*)

Remove a [Formation](#) by id from the [Population](#) and the corresponding [GroupSocialSpace](#).

Parameters

<i>formation↔ Id</i>	: Formation id to remove from the Population
--------------------------	--

Returns

0 on success, 0 otherwise (if the [Formation](#) does not exist)

5.17.3.11 void Population::setFormations (const std::vector< [Formation](#) * > & *formations*)

Simpler setter.

Parameters

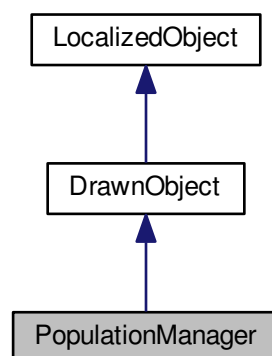
<i>formations</i>	
-------------------	--

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

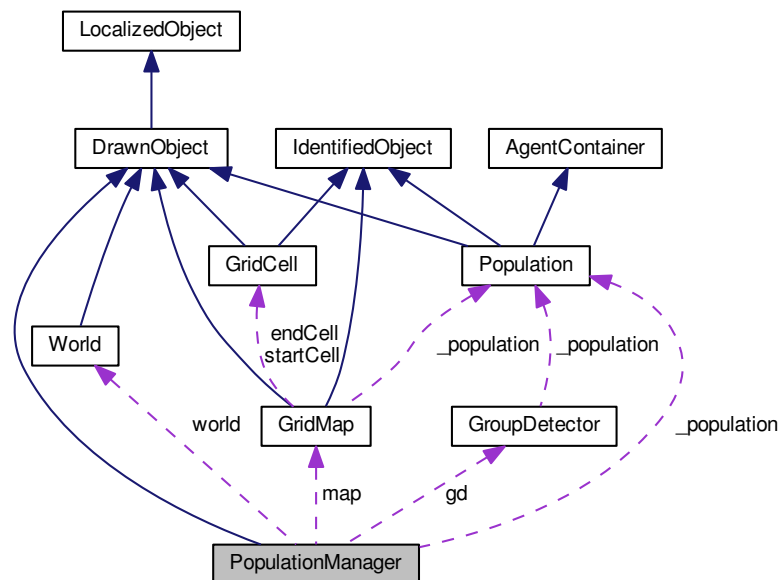
- [src/agentManagement/Population.h](#)
- [src/agentManagement/Population.cpp](#)

5.18 PopulationManager Class Reference

Inheritance diagram for PopulationManager:



Collaboration diagram for PopulationManager:



Public Member Functions

- **PopulationManager** ([World](#) *world)
- **PopulationManager** ([World](#) *world, std::string feature_file, Vector3d p=Vector3d())
- **PopulationManager** (std::string feature_file, std::string gt_file, Vector3d p=Vector3d())
- int **loadFrame** (unsigned int flndex)
- int **loadJson** ()
- int **loadFeatureJson** ()
- int **loadGroundTruthJson** ()
- void **findDataBounds** ()
- [Formation](#) * **getHighestFormationInteractionPotential** ()
- void **runTest** ()
- int **nextFrame** ()
- int **previousFrame** ()
- void **findInteraction** ()
- void **draw** ([World](#) *world)
- void **update** ()
- [Population](#) * **getPopulation** () const
- void **setPopulation** ([Population](#) *population)
- const std::string & **getFeatureFile** () const
- void **setFeatureFile** (const std::string &featureFile)
- const ofxJSONElement & **getFeatures** () const
- void **setFeatures** (const ofxJSONElement &features)
- unsigned int **getFrameIndex** () const
- void **setFrameIndex** (unsigned int frameIndex)
- const ofxJSONElement & **getGroundTruth** () const
- void **setGroundTruth** (const ofxJSONElement &groundTruth)
- bool **isGtEnabled** () const

- void **setGtEnabled** (bool gtEnabled)
- const std::string & **getGtFile** () const
- void **setGtFile** (const std::string >File)
- bool **isLoading** () const
- void **setLoaded** (bool loaded)
- [World](#) * **getWorld** () const
- [GridMap](#) * **getMap** () const

Protected Attributes

- [Population](#) * **_population**
- std::vector< [Formation](#) * > **_GT**
- ofxJSONElement **features**
- ofxJSONElement **groundTruth**
- std::string **feature_file**
- std::string **gt_file**
- [World](#) * **world**
- [GridMap](#) * **map**
- [GroupDetector](#) * **gd**
- double **min_x**
- double **min_y**
- double **max_x**
- double **max_y**
- bool **loaded** = 0
- bool **gt_enabled** = 1
- unsigned int **frameIndex** = 0

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

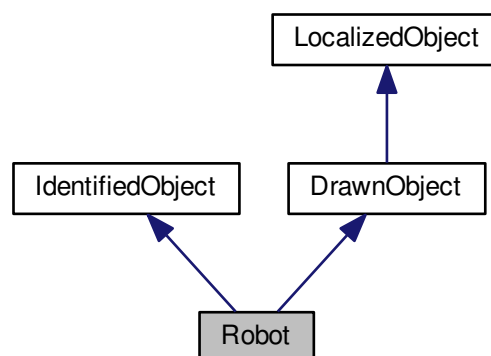
- src/agentManagement/PopulationManager.h
- src/agentManagement/PopulationManager.cpp

5.19 Robot Class Reference

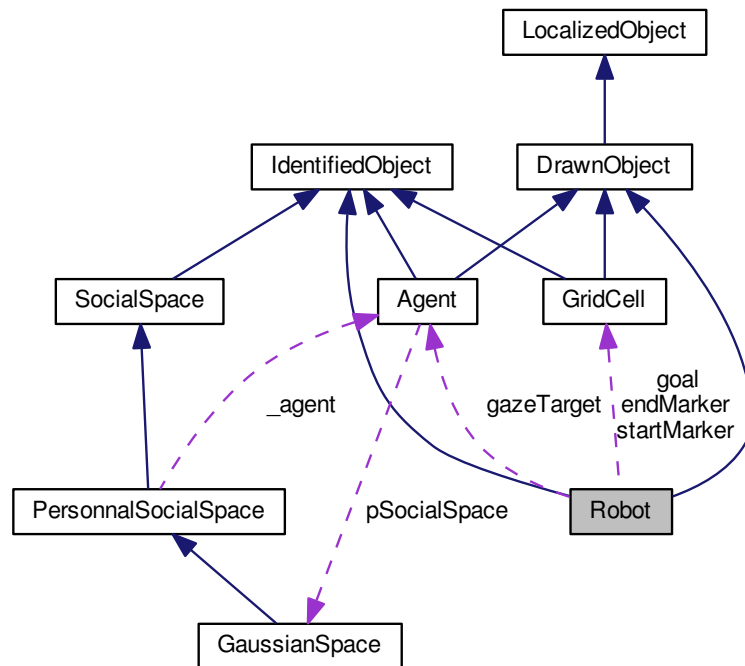
This class represent the [Robot](#).

```
#include <Robot.h>
```

Inheritance diagram for Robot:



Collaboration diagram for Robot:



Public Member Functions

- **Robot** (Vector3d **position**=Vector3d(0, 0, 0), double **theta**=0)
Constructor.
- **~Robot** ()
Destructor.
- void **update** ()
*Compute the movement and the rotation of the **Robot** based on the computed path and linear interpolation between points.*
- void **resetPathFinding** ()
Reset path finding process.
- **GridCell** * **getGoal** () const
Simple getter.
- void **setGoal** (**GridCell** ***goal**)
Simple setter.
- std::vector< **GridCell** * > **getPath** () const
Simple getter.
- void **setPath** (std::vector< **GridCell** * > **path**)
Simple setter.

Public Attributes

- `GridCell * goal`
The current goal of the [Robot](#).
- `std::vector< GridCell * > path`
The current path followed by the [Robot](#).
- `int pathIndex`
The current path index.
- `GridCell * startMarker`
The starting [GridCell](#).
- `GridCell * endMarker`
The goal [GridCell](#).
- `std::chrono::time_point< std::chrono::system_clock > startMoveTime`
Start time of the movement.
- `std::chrono::time_point< std::chrono::system_clock > startRotTime`
Start time of the rotation.
- `Agent * gazeTarget`
The [Robot](#) gaze targeted [Agent](#) for social signals.
- `double targetAngle`
The current targeted angle.
- `double * finalTargetAngle`
The targeted angle at the end of the movement.
- `double startAngle`
The initial angle of the [Robot](#).
- `double rotDist`
The rotation distance of the actual segment.
- `double moveDist`
The movement distance of the actual segment.
- `double alphaMove = 0.05`
Used by the movement formulae.
- `double alphaRot = 0.05`
Used by the rotation formulae.
- `double moveSpeed = 0.5f`
The [Robot](#) move speed in meter/s.
- `double rotSpeed = 4.0f`
The [Robot](#) move speed in radian/s.
- `bool initPoint = 1`
Enable or disable path finding segment initialization.

Additional Inherited Members

5.19.1 Detailed Description

This class represent the [Robot](#).

This class manage the [Robot](#) and its movement

5.19.2 Constructor & Destructor Documentation

5.19.2.1 `Robot::Robot (Vector3d position = Vector3d(0, 0, 0), double theta = 0)`

Constructor.

Constructor of the [Robot](#) class

Parameters

<i>position</i>	: Initial position of the Robot in World
<i>theta</i>	: Initial angle of the Robot

5.19.2.2 `Robot::~~Robot ()`

Destructor.

Destructor of the [Robot](#) class

5.19.3 Member Function Documentation

5.19.3.1 `GridCell * Robot::getGoal () const`

Simple getter.

Returns

goal

5.19.3.2 `std::vector< GridCell * > Robot::getPath () const`

Simple getter.

Returns

path

5.19.3.3 `void Robot::setGoal (GridCell * goal)`

Simple setter.

Parameters

<i>goal</i>	
-------------	--

5.19.3.4 `void Robot::setPath (std::vector< GridCell * > path)`

Simple setter.

Parameters

<i>path</i>	
-------------	--

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

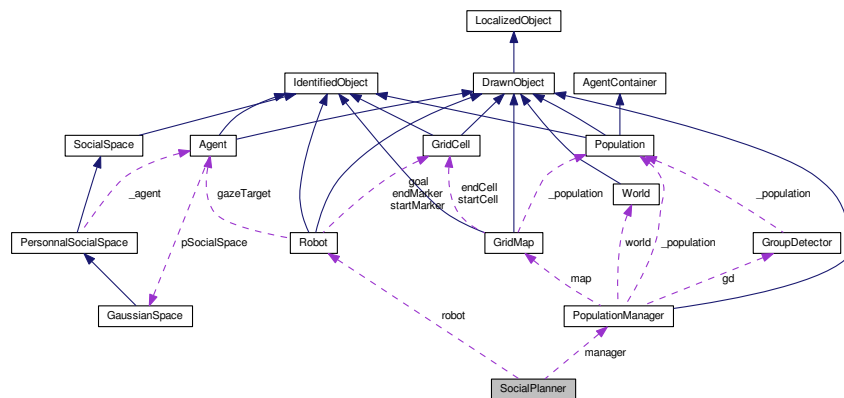
- src/agentManagement/[Robot.h](#)
- src/agentManagement/[Robot.cpp](#)

5.20 SocialPlanner Class Reference

This class is a state machine controlling the [Robot](#) behavior.

```
#include <SocialPlanner.h>
```

Collaboration diagram for SocialPlanner:



Public Types

- enum [SocialState](#) { **SEEK_INTERACTION**, **ENGAGEMENT**, **INTERACTION**, **DISENGAGE** }
- The states available for the state machine.*

Public Member Functions

- [SocialPlanner](#) ([PopulationManager](#) *popManager, [Robot](#) *robot)
Constructor.
- virtual [~SocialPlanner](#) ()
Destructor.
- void [update](#) ()
Update the state of the robot behavior,.
- void [seek_interaction](#) ()
Seek interaction state.
- void [engage](#) ()

- Engage state.*
- void [interact](#) ()
- Interaction state.*
- void [disengage](#) ()
- Disengage state.*
- [PopulationManager](#) * [getManager](#) () const
- Simple getter.*
- [Robot](#) * [getRobot](#) () const
- Simple getter.*

Public Attributes

- bool [interactionStarted](#) = 0
- Interaction state started.*
- bool [seekStarted](#) = 0
- Seek interaction state started.*
- std::chrono::time_point< std::chrono::system_clock > [startInteractionTime](#)
- Time when interaction state started.*
- std::chrono::time_point< std::chrono::system_clock > [startSeekTime](#)
- Time when seek interaction state started.*
- std::chrono::time_point< std::chrono::system_clock > [startMutualFacialGaze](#)
- Time when mutual facial gaze started.*
- [SocialState](#) [state](#) = SEEK_INTERACTION
- Actual state, default is SEEK_INTERACTION.*
- [PopulationManager](#) * [manager](#)
- The [PopulationManager](#) used by the [SocialPlanner](#).*
- [Robot](#) * [robot](#)
- The [Robot](#) controlled by the [SocialPlanner](#).*
- Vector3d [savedPosition](#)
- Interaction position saved, dirty.*
- int [gazeTargetIndex](#) = 0
- [Agent](#) index for the gaze target.*

5.20.1 Detailed Description

This class is a state machine controlling the [Robot](#) behavior.

Todo This class should be an interface to implement different behavior for the robot

5.20.2 Constructor & Destructor Documentation

5.20.2.1 [SocialPlanner::SocialPlanner](#) ([PopulationManager](#) * *popManager*, [Robot](#) * *robot*)

Constructor.

Constructor of the [SocialPlanner](#) class

Parameters

<i>popManager</i>	: The Population where the robot evolves
<i>robot</i>	: The Robot controlled by this behavior

5.20.2.2 SocialPlanner::~SocialPlanner () [virtual]

Destructor.

Destructor of the [SocialPlanner](#) class

5.20.3 Member Function Documentation

5.20.3.1 void SocialPlanner::disengage ()

Disengage state.

The robot ends the interaction by going back to the seek interaction state

Todo Execute some social signals for disengagement

5.20.3.2 void SocialPlanner::engage ()

Engage state.

The robot move to the interaction position related to the [Formation](#) and process some social signal like mutual facial gaze to communicate its intentions. And switch to the interaction state when the goal is reached

5.20.3.3 PopulationManager * SocialPlanner::getManager () const

Simple getter.

Returns

[PopulationManager](#)

5.20.3.4 Robot * SocialPlanner::getRobot () const

Simple getter.

Returns

[Robot](#)

5.20.3.5 void SocialPlanner::interact ()

Interaction state.

The robot simulate an interaction with the [Formation](#) for INTERACTION_MAX_TIME, then switch to the disengage state.

5.20.3.6 void SocialPlanner::seek_interaction ()

Seek interaction state.

The robot is looking for interaction, it goes randomly around the [GridMap](#) and after SEEK_INTERACTION_MIN_TIME it looks for the highest interaction potential available in the formations superior to INTERACTION_POTENTIAL_THRESHOLD then switch to the engage state

5.20.3.7 void SocialPlanner::update ()

Update the state of the robot behavior,.

Update the state of the robot behavior, by executing the function related to its actual state

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

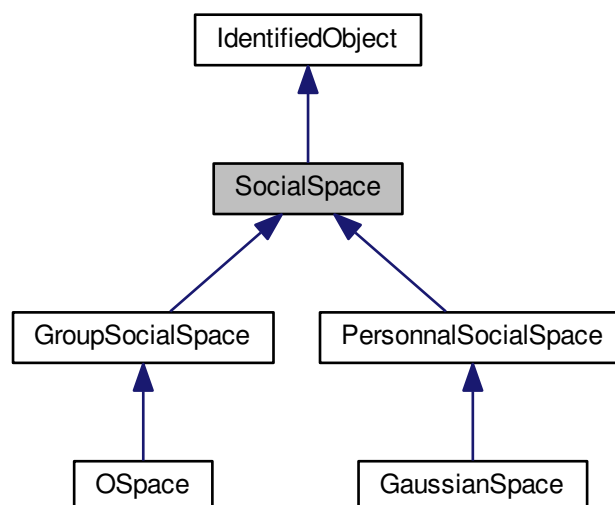
- [src/socialProcessing/SocialPlanner.h](#)
- [src/socialProcessing/SocialPlanner.cpp](#)

5.21 SocialSpace Class Reference

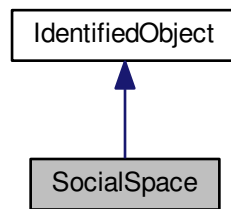
This class is an abstract class for representing [SocialSpace](#).

```
#include <SocialSpace.h>
```

Inheritance diagram for SocialSpace:



Collaboration diagram for SocialSpace:



Public Member Functions

- [SocialSpace](#) (int `id`=0)
Constructor.
- virtual [~SocialSpace](#) ()
Destructor.

Additional Inherited Members

5.21.1 Detailed Description

This class is an abstract class for representing [SocialSpace](#).

5.21.2 Constructor & Destructor Documentation

5.21.2.1 `SocialSpace::SocialSpace (int id = 0)`

Constructor.

Constructor of the [SocialSpace](#) class

Parameters

<code><i>id</i></code>	: The unique identifier of the SocialSpace
------------------------	--

5.21.2.2 `SocialSpace::~~SocialSpace ()` [virtual]

Destructor.

Destructor of the [SocialSpace](#) class

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

- int [do_read](#) ()
Read input data on the receive socket.
- int [do_send](#) (uint8_t *sendBuffer, int sendBuffer_size)
Send output data on the send socket.
- void [update](#) ()
Parse the received data and send every frame.
- void [run](#) ()
The threaded read socket.
- void [sendAll](#) ()
Send all the frame.
- void [updateOrPushAgent](#) (int id, float x, float y, float z, float theta)
Update or add [Agent](#) to the [Population](#).

Protected Attributes

- int [portNumber](#)
The port used by the UDP Server.
- struct sockaddr_in [myAddr](#)
The network information of the server.
- struct sockaddr_in [fromAddr](#)
The network information of the client.
- int [udpReceiveSocket](#)
Receive socket file descriptor.
- int [udpSendSocket](#)
Send socket file descriptor.
- uint8_t [recvBuffer](#) [[recvBuffer_size](#)]
Receive buffer.
- [SocialPlanner](#) * [planner](#)
The related [SocialPlanner](#).

Static Protected Attributes

- static const int [recvBuffer_size](#) = 122
Receive buffer size.

5.22.1 Detailed Description

This class manage the UDP Server sending computed data to a visualization software and receiving [Agent](#) data from other sensor sources.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 UDPServer::UDPServer (int port, SocialPlanner * sPlanner)

Constructor.

Constructor of the [UDPServer](#) class

Parameters

<i>port</i>	: The server port
<i>sPlanner</i>	: The SocialPlanner connected to this server

5.22.2.2 UDPServer::~UDPServer () [virtual]

Destructor.

Destructor of the [UDPServer](#) class

5.22.3 Member Function Documentation

5.22.3.1 int UDPServer::do_read ()

Read input data on the receive socket.

Returns

number of Bytes read, 0 or negative value in case of I/O error

5.22.3.2 int UDPServer::do_send (uint8_t * *sendBuffer*, int *sendBuffer_size*)

Send output data on the send socket.

Parameters

<i>sendBuffer</i>	: The data buffer
<i>sendBuffer_size</i>	: The size of the data buffer

Returns

The number of bytes send, 0 or negative value in case of I/O error

5.22.3.3 int UDPServer::parse ()

Parse the data received by the receive thread.

Returns

A positive number if parse was successful, 0 otherwise

5.22.3.4 int UDPServer::parse_frame0 ()

Parse frame0 containing [Agent](#) data.

Returns

The number of [Agent](#) parsed

5.22.3.5 int UDPServer::send_frame0 ()

Create frame0 containing all [Agent](#) data from the [Population](#).

Returns

0

5.22.3.6 int UDPServer::send_frame1 ()

Create frame1 containing the [Robot](#) data.

Returns

0

5.22.3.7 int UDPServer::send_frame2 ()

Create frame2 containing [Robot](#) path.

Returns

0

5.22.3.8 int UDPServer::send_frame3 ()

Create frame3 containing [Formation](#) data.

Returns

0

5.22.3.9 std::thread UDPServer::spawn ()

Launch the [UDPServer](#) receive function in a new thread.

Returns

The thread created

5.22.3.10 void UDPServer::updateOrPushAgent (int *id*, float *x*, float *y*, float *z*, float *theta*)

Update or add [Agent](#) to the [Population](#).

Parameters

<i>id</i>	: The unique identifier of the Agent
<i>x</i>	: The x coordinate of the Agent
<i>y</i>	: The y coordinate of the Agent
<i>z</i>	: The z coordinate of the Agent
<i>theta</i>	: The angle of the Agent

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

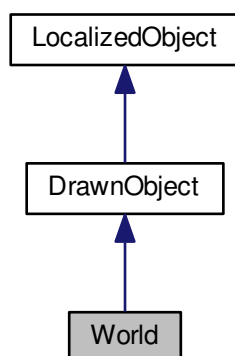
- [src/UDPServer.h](#)
- [src/UDPServer.cpp](#)

5.23 World Class Reference

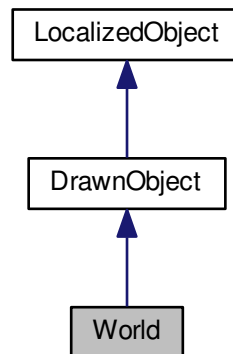
This class represent the main frame coordinates and its projection in pixels.

```
#include <World.h>
```

Inheritance diagram for World:



Collaboration diagram for World:



Public Member Functions

- **World** (double **width**, double **height**, int **widthView**, int **heightView**, Vector3d **position**=Vector3d(), double **theta**=0)
Constructor.
- virtual **~World** ()
Destructor.
- double **getHeight** () const
Simple getter.
- void **setHeight** (double **height**)
Simple setter.
- double **getHeightView** () const
Simple getter.
- void **setHeightView** (double **heightView**)
Simple setter.
- double **getWidth** () const
Simple getter.
- void **setWidth** (double **width**)
Simple setter.
- double **getWidthView** () const
Simple getter.
- void **setWidthView** (double **widthView**)
Simple setter.

Public Attributes

- double **width**
*The real width of the **World** in meter.*
- double **height**
*The real height of the **World** in meter.*
- double **widthView**
*The projection width of the **World** in pixel.*
- double **heightView**
*The projection height of the **World** in pixel.*

Additional Inherited Members

5.23.1 Detailed Description

This class represent the main frame coordinates and its projection in pixels.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 `World::World (double width, double height, int widthView, int heightView, Vector3d position = Vector3d(), double theta = 0)`

Constructor.

Constructor of the [World](#) class

Parameters

<i>width</i>	: The real width of the World in meter
<i>height</i>	: The real height of the World in meter
<i>widthView</i>	: The projection width of the World in pixel
<i>heightView</i>	: The projection height of the World in pixel
<i>position</i>	: The position of the World in the GUI in pixel
<i>theta</i>	: The rotation of the World in the GUI in radian

5.23.2.2 `World::~~World () [virtual]`

Destructor.

Destructor of the [World](#) class

5.23.3 Member Function Documentation

5.23.3.1 `double World::getHeight () const`

Simple getter.

Returns

height

5.23.3.2 `double World::getHeightView () const`

Simple getter.

Returns

heightView

5.23.3.3 double World::getWidth () const

Simple getter.

Returns

width

5.23.3.4 double World::getWidthView () const

Simple getter.

Returns

widthView

5.23.3.5 void World::setHeight (double *height*)

Simple setter.

Parameters

<i>height</i>	
---------------	--

5.23.3.6 void World::setHeightView (double *heightView*)

Simple setter.

Parameters

<i>heightView</i>	
-------------------	--

5.23.3.7 void World::setWidth (double *width*)

Simple setter.

Parameters

<i>width</i>	
--------------	--

5.23.3.8 void World::setWidthView (double *widthView*)

Simple setter.

Parameters

<i>widthView</i>	
------------------	--

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

- src/worldRepresentation/[World.h](#)
- src/worldRepresentation/World.cpp

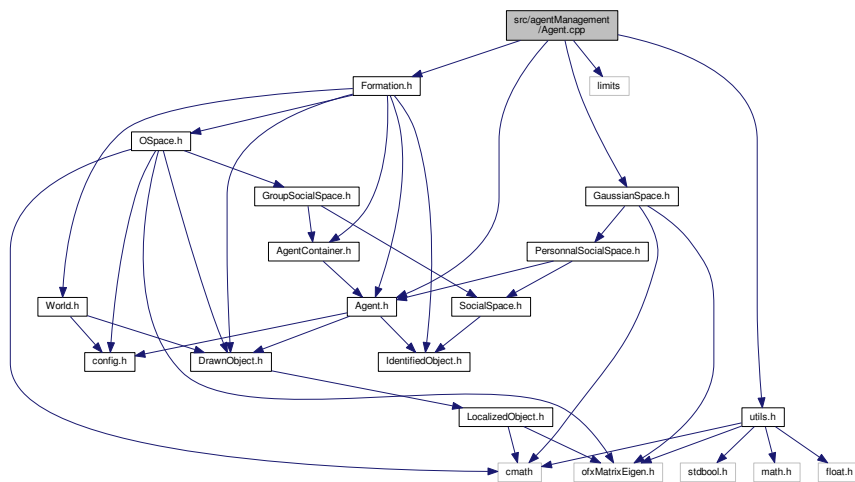
Chapter 6

File Documentation

6.1 src/agentManagement/Agent.cpp File Reference

```
#include "Agent.h"  
#include "GaussianSpace.h"  
#include "Formation.h"  
#include <limits>  
#include "utils.h"
```

Include dependency graph for Agent.cpp:



6.1.1 Detailed Description

Author

Paco Dupont

Version

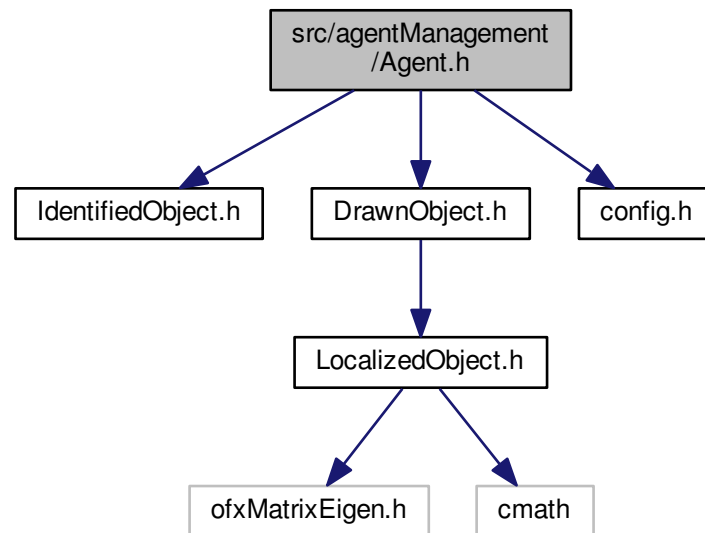
0.1

Date

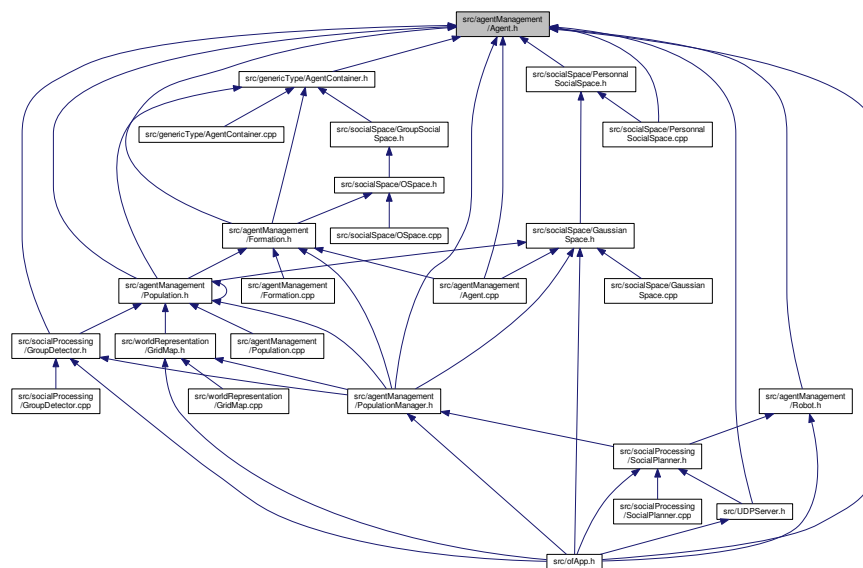
24 mars 2017

6.2 src/agentManagement/Agent.h File Reference

```
#include "IdentifiedObject.h"
#include "DrawnObject.h"
#include "config.h"
Include dependency graph for Agent.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Agent](#)

This class represent the Agents.

6.2.1 Detailed Description

Author

Paco Dupont

Version

0.1

Date

24 mars 2017

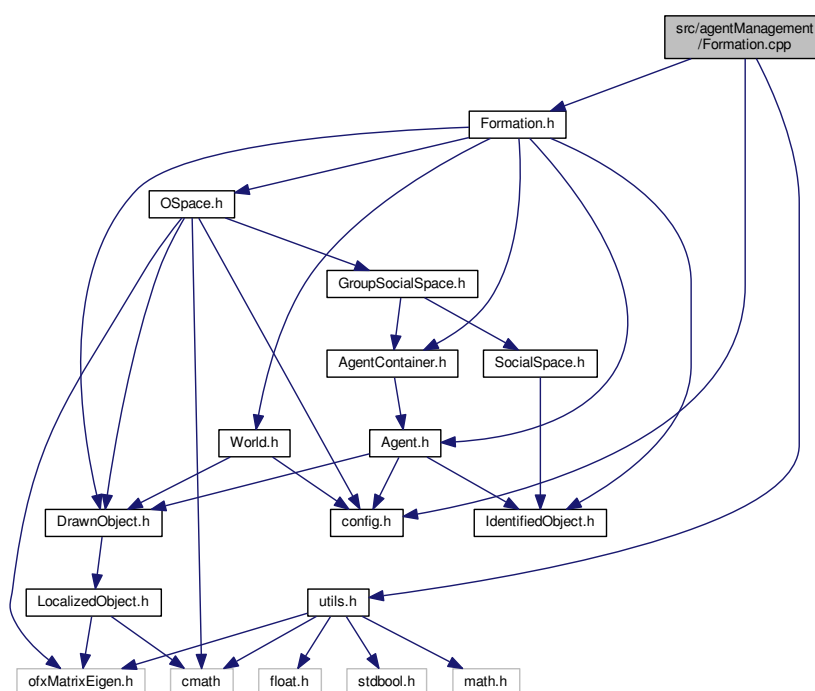
6.3 src/agentManagement/Formation.cpp File Reference

```
#include "Formation.h"
```

```
#include "config.h"
```

```
#include "utils.h"
```

Include dependency graph for Formation.cpp:



6.3.1 Detailed Description

Author

Paco Dupont

Version

0.1

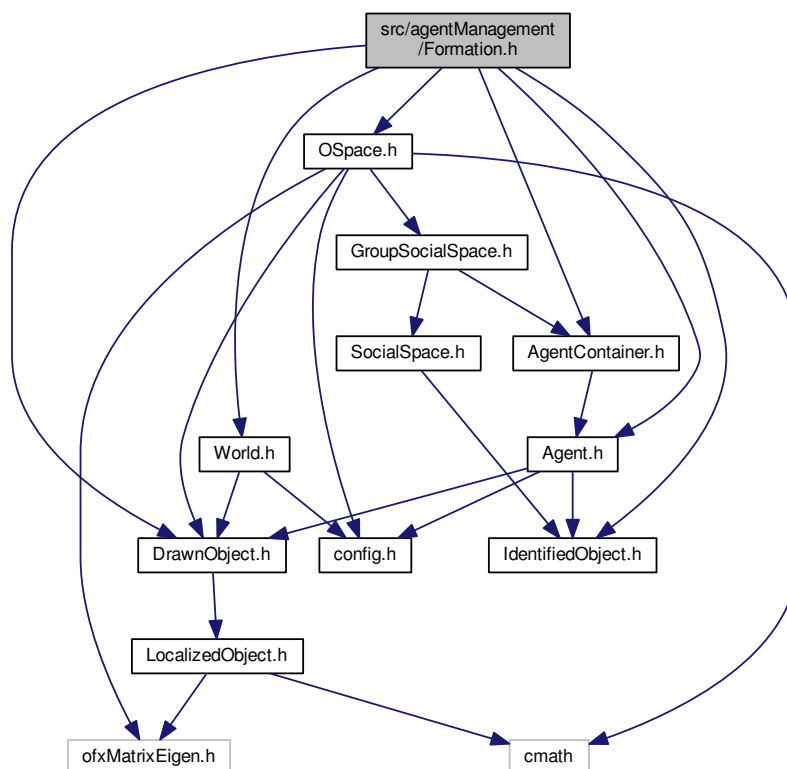
Date

24 mars 2017

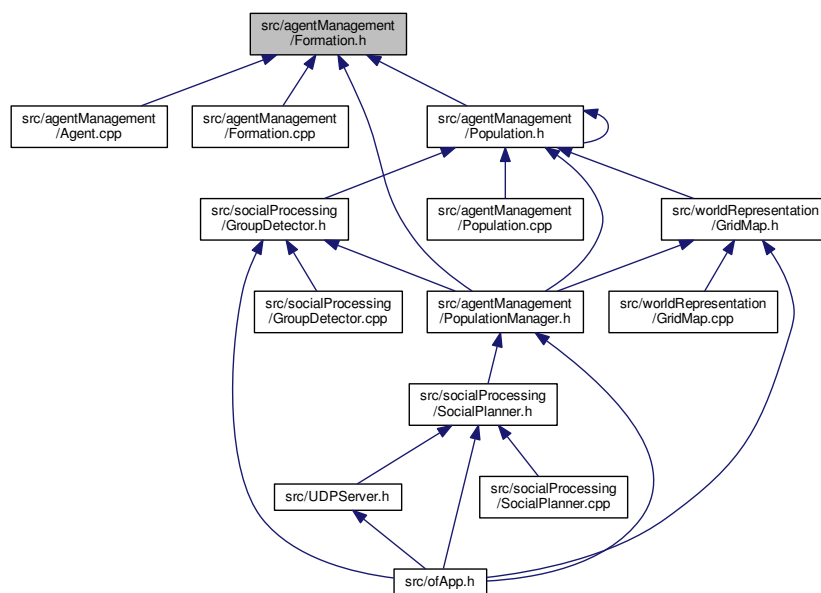
6.4 src/agentManagement/Formation.h File Reference

```
#include "Agent.h"
#include "IdentifiedObject.h"
#include "DrawnObject.h"
#include "AgentContainer.h"
#include "OSpace.h"
#include "World.h"
```

Include dependency graph for Formation.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Formation](#)

This class represent the social [Formation](#).

6.4.1 Detailed Description

Author

Paco Dupont

Version

0.1

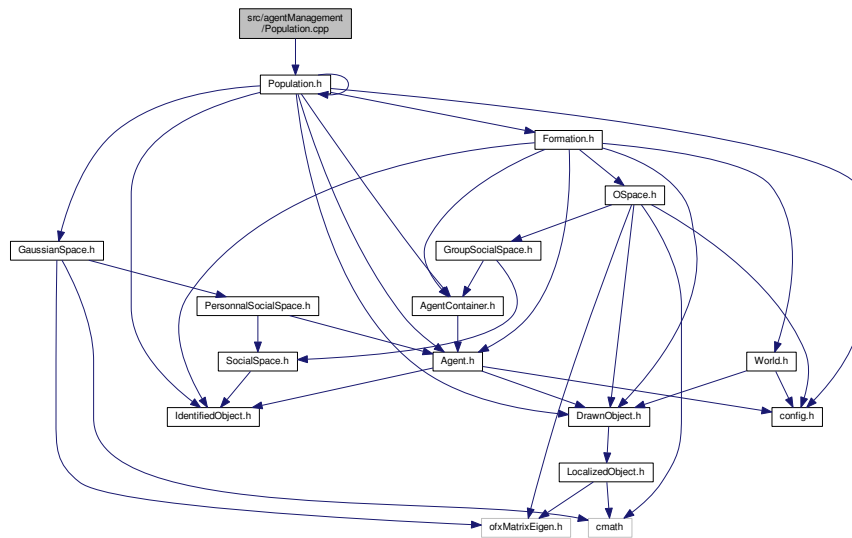
Date

24 mars 2017

6.5 src/agentManagement/Population.cpp File Reference

```
#include "Population.h"
```

Include dependency graph for Population.cpp:



6.5.1 Detailed Description

Author

Paco Dupont

Version

0.1

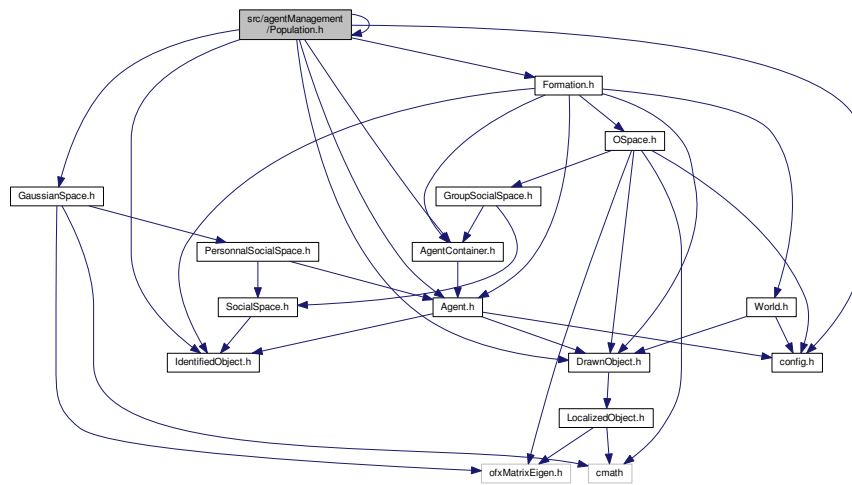
Date

24 mars 2017

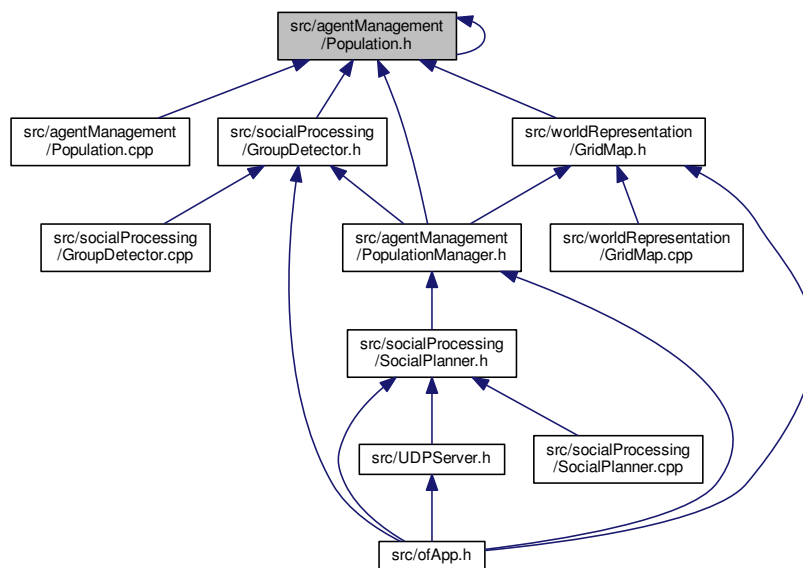
6.6 src/agentManagement/Population.h File Reference

```
#include "Agent.h"
#include "Formation.h"
#include "IdentifiedObject.h"
#include "DrawnObject.h"
#include "AgentContainer.h"
#include "Population.h"
#include "GaussianSpace.h"
#include "config.h"
```

Include dependency graph for Population.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Population](#)

This class represent [Population](#) around the [Robot](#).

6.6.1 Detailed Description

Author

Paco Dupont

Version

0.1

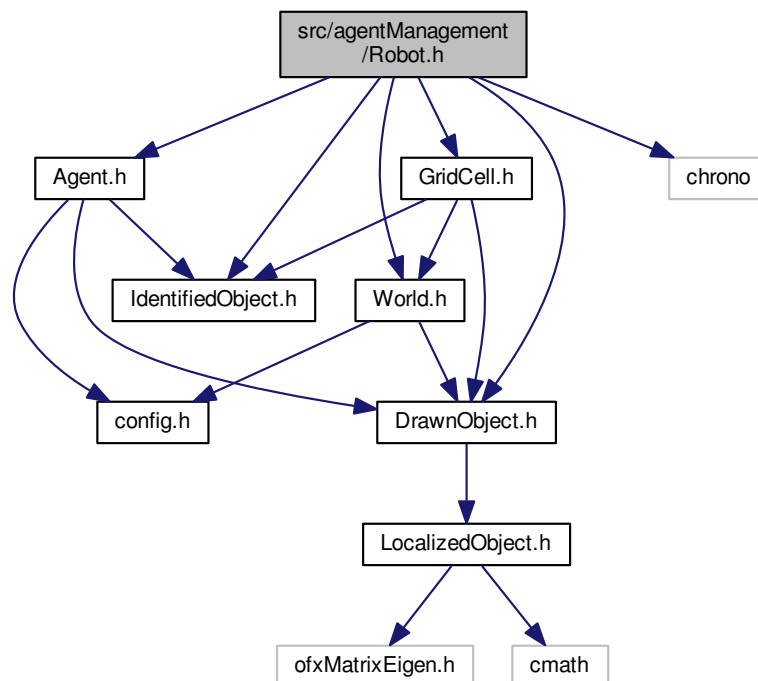
Date

24 mars 2017

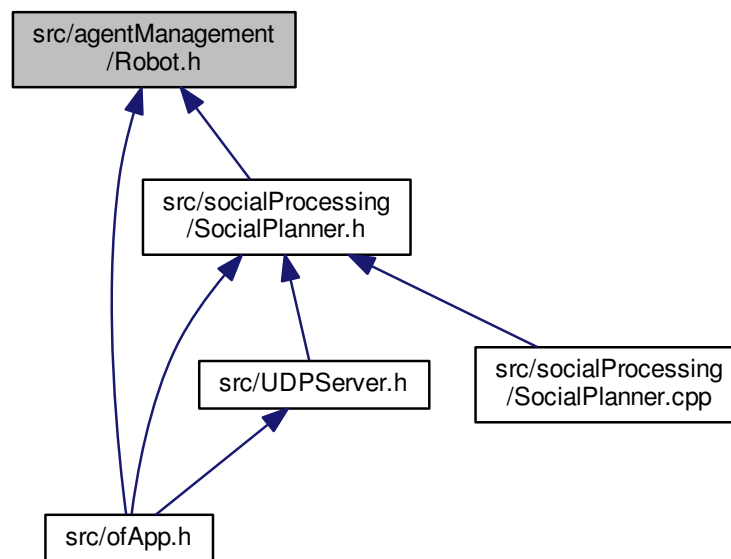
6.7 src/agentManagement/Robot.h File Reference

```
#include "IdentifiedObject.h"  
#include "DrawnObject.h"  
#include "World.h"  
#include "GridCell.h"  
#include "Agent.h"  
#include <chrono>
```

Include dependency graph for Robot.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Robot](#)

This class represent the [Robot](#).

6.7.1 Detailed Description

Author

Paco Dupont

Version

0.1

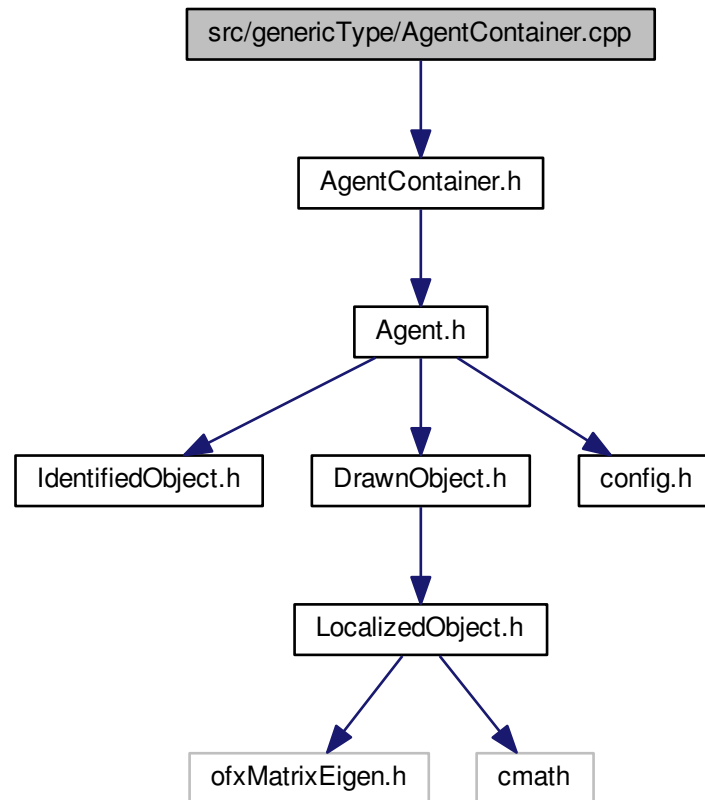
Date

7 juin 2017

6.8 src/genericType/AgentContainer.cpp File Reference

```
#include "AgentContainer.h"
```

Include dependency graph for AgentContainer.cpp:



6.8.1 Detailed Description

Author

Paco Dupont

Version

0.1

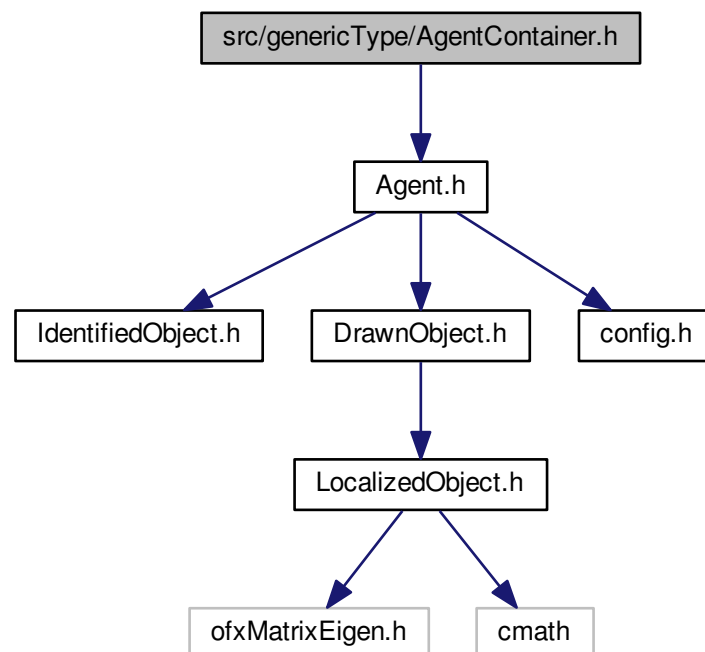
Date

27 mars 2017

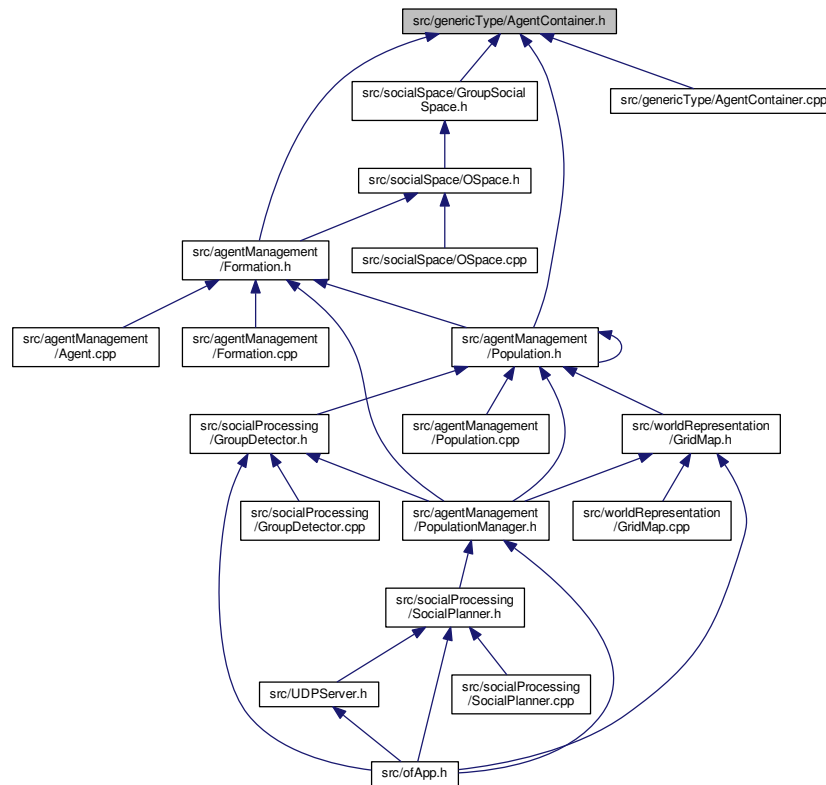
6.9 src/genericType/AgentContainer.h File Reference

```
#include "Agent.h"
```

Include dependency graph for AgentContainer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [AgentContainer](#)

This class is an interface for class that contains multiples Agents.

6.9.1 Detailed Description

Author

Paco Dupont

Version

0.1

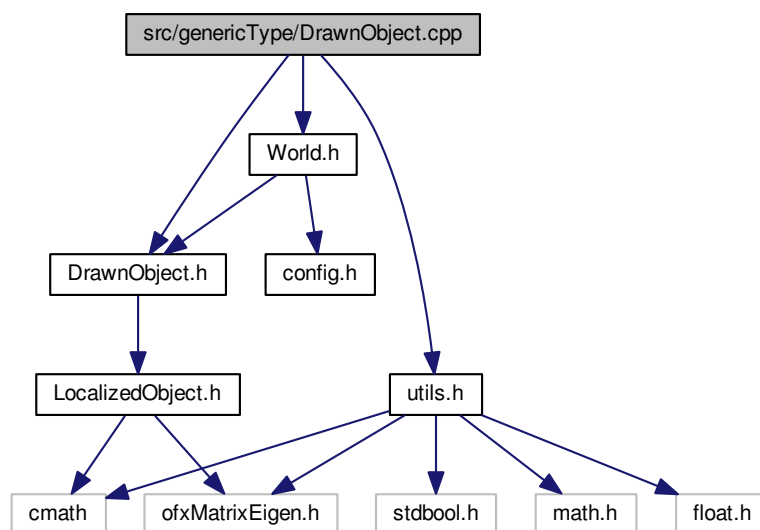
Date

27 mars 2017

6.10 src/genericType/DrawnObject.cpp File Reference

```
#include "DrawnObject.h"  
#include "World.h"  
#include "utils.h"
```

Include dependency graph for DrawnObject.cpp:



6.10.1 Detailed Description

Author

Paco Dupont

Version

0.1

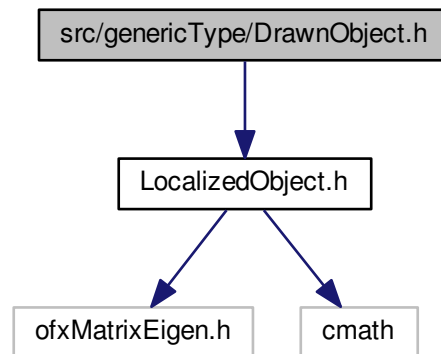
Date

27 mars 2017

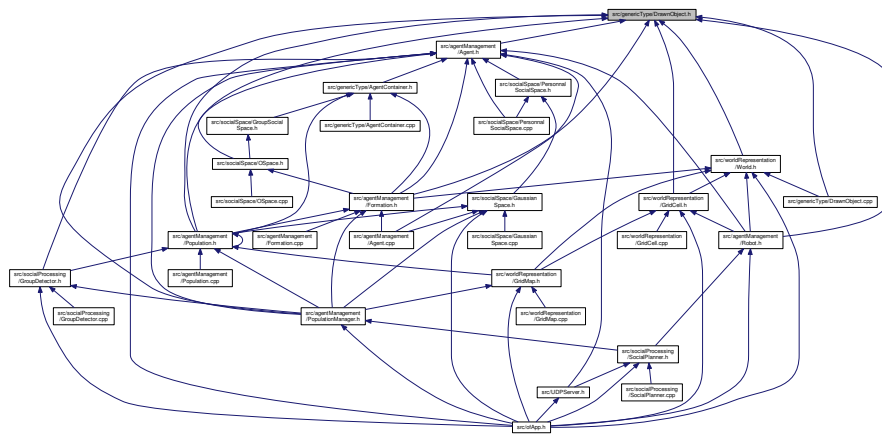
6.11 src/genericType/DrawnObject.h File Reference

```
#include "LocalizedObject.h"
```

Include dependency graph for DrawnObject.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [DrawnObject](#)

This class is an interface for class that are drawn on OFX gui.

6.11.1 Detailed Description

Author

Paco Dupont

Version

0.1

Date

27 mars 2017

Classes

- class [IdentifiedObject](#)

This class is an interface for class that need a unique identifier.

6.13.1 Detailed Description

Author

Paco Dupont

Version

0.1

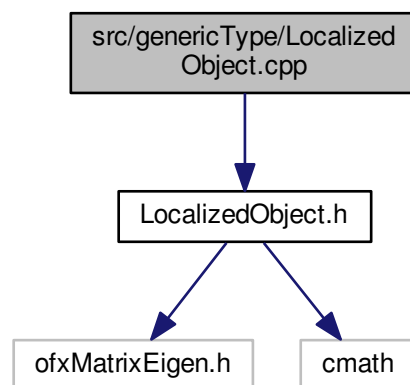
Date

27 mars 2017

6.14 src/genericType/LocalizedObject.cpp File Reference

```
#include "LocalizedObject.h"
```

Include dependency graph for LocalizedObject.cpp:



6.14.1 Detailed Description

Author

Paco Dupont

Version

0.1

Date

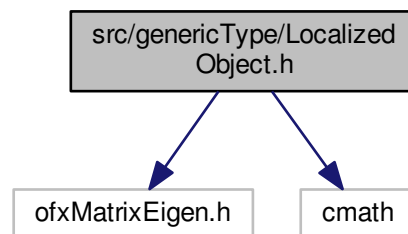
27 mars 2017

6.15 src/genericType/LocalizedObject.h File Reference

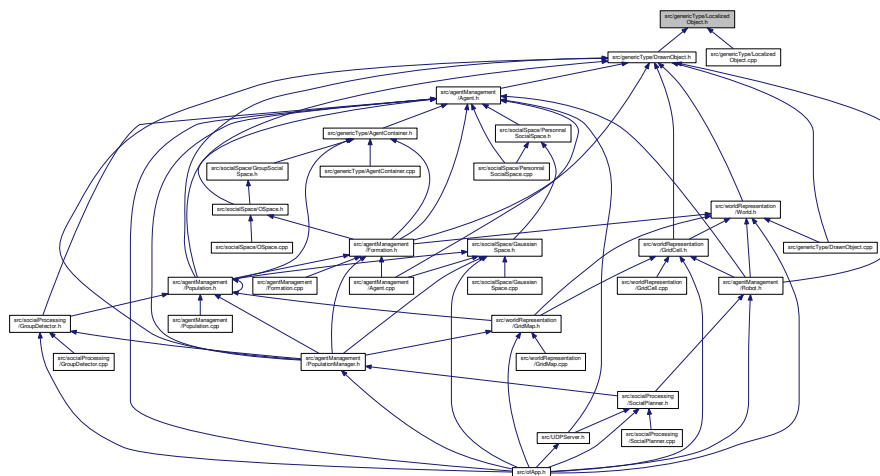
```
#include "ofxMatrixEigen.h"
```

```
#include <cmath>
```

Include dependency graph for LocalizedObject.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [LocalizedObject](#)

This class is an interface for object that are localized in real [World](#).

6.15.1 Detailed Description

Author

Paco Dupont

Version

0.1

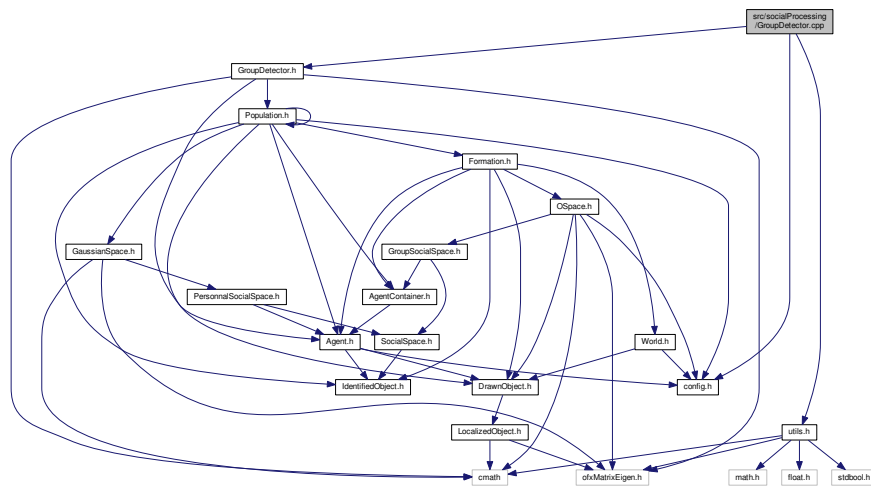
Date

27 mars 2017

6.16 src/socialProcessing/GroupDetector.cpp File Reference

```
#include "GroupDetector.h"  
#include "utils.h"  
#include "config.h"
```

Include dependency graph for GroupDetector.cpp:



6.16.1 Detailed Description

Author

Paco Dupont

Version

0.1

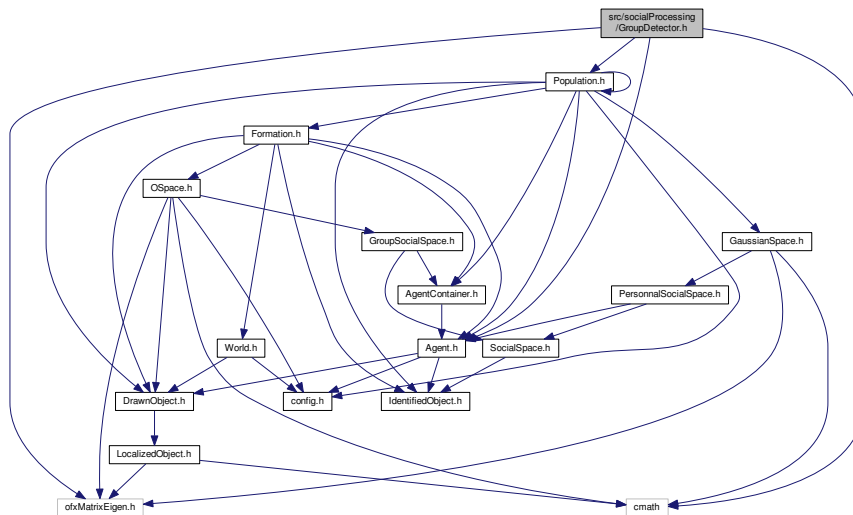
Date

10 avril 2017

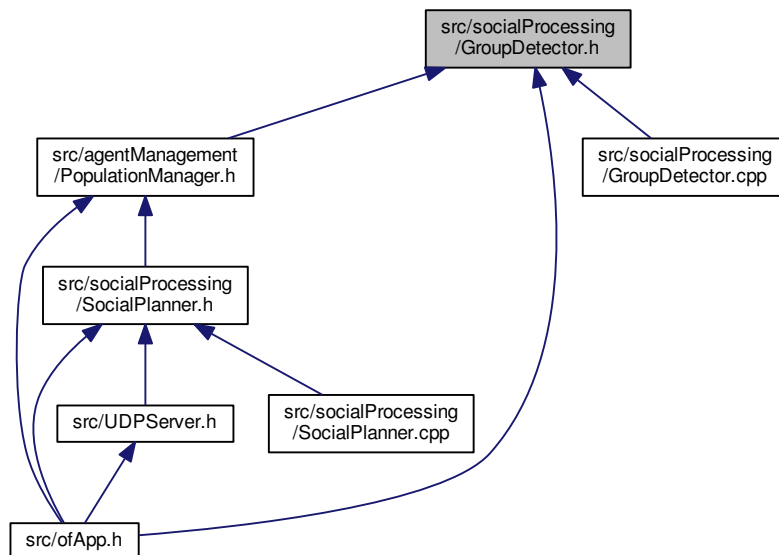
6.17 src/socialProcessing/GroupDetector.h File Reference

```
#include "Population.h"
#include "Agent.h"
#include "ofxMatrixEigen.h"
#include <cmath>
```

Include dependency graph for GroupDetector.h:



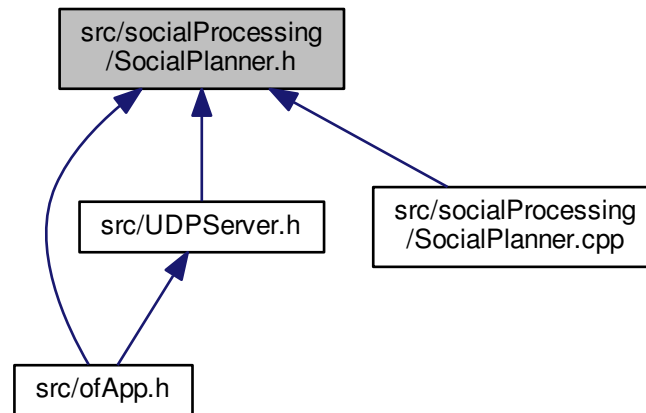
This graph shows which files directly or indirectly include this file:



Classes

- class [GroupDetector](#)

This graph shows which files directly or indirectly include this file:



Classes

- class [SocialPlanner](#)

This class is a state machine controlling the [Robot](#) behavior.

6.19.1 Detailed Description

Author

Paco Dupont

Version

0.1

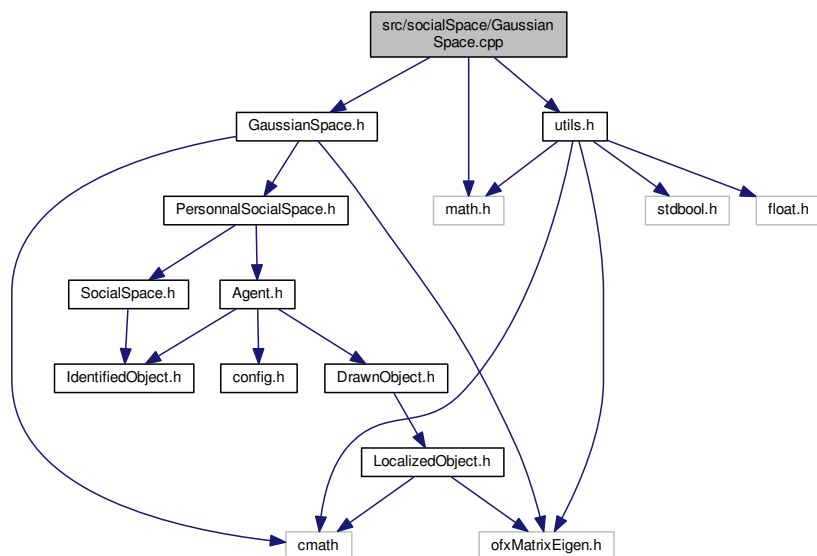
Date

7 juin 2017

6.20 src/socialSpace/GaussianSpace.cpp File Reference

```
#include "GaussianSpace.h"
#include "math.h"
#include "utils.h"
```

Include dependency graph for GaussianSpace.cpp:



6.20.1 Detailed Description

Author

Paco Dupont

Version

0.1

Date

28 mars 2017

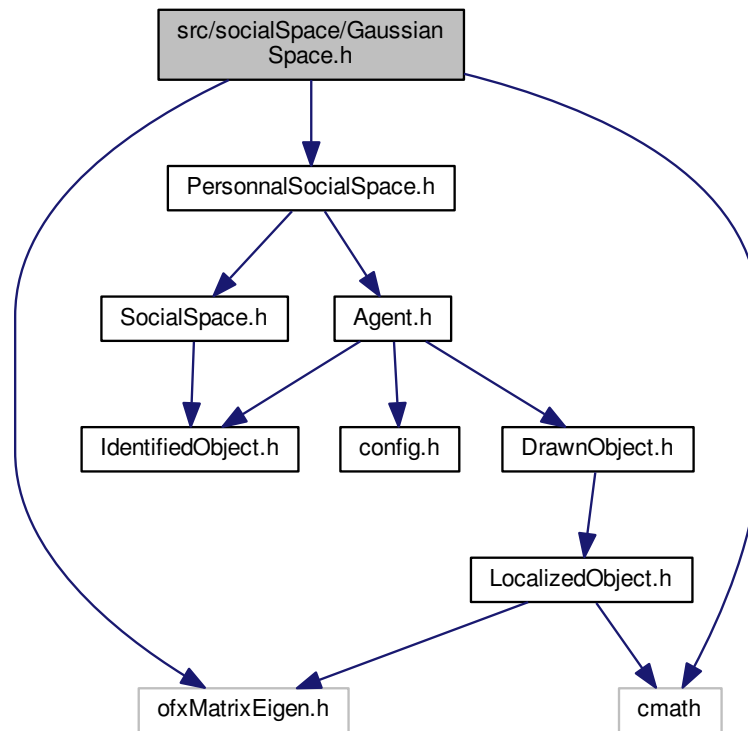
6.21 src/socialSpace/GaussianSpace.h File Reference

```

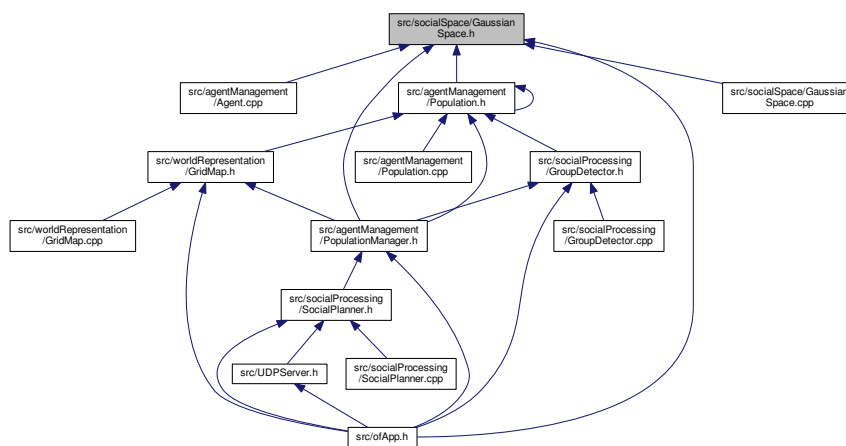
#include "PersonalSocialSpace.h"
#include "ofxMatrixEigen.h"
#include <cmath>

```

Include dependency graph for GaussianSpace.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [GaussianSpace](#)

This class is an implementation of the [PersonnalSocialSpace](#).

6.21.1 Detailed Description

Author

Paco Dupont

Version

0.1

Date

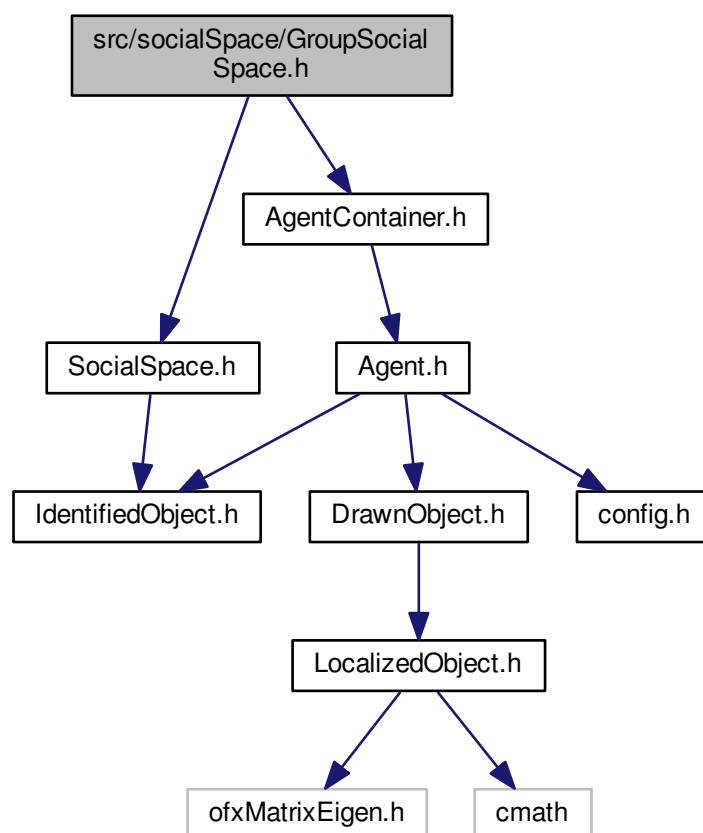
28 mars 2017

6.22 src/socialSpace/GroupSocialSpace.h File Reference

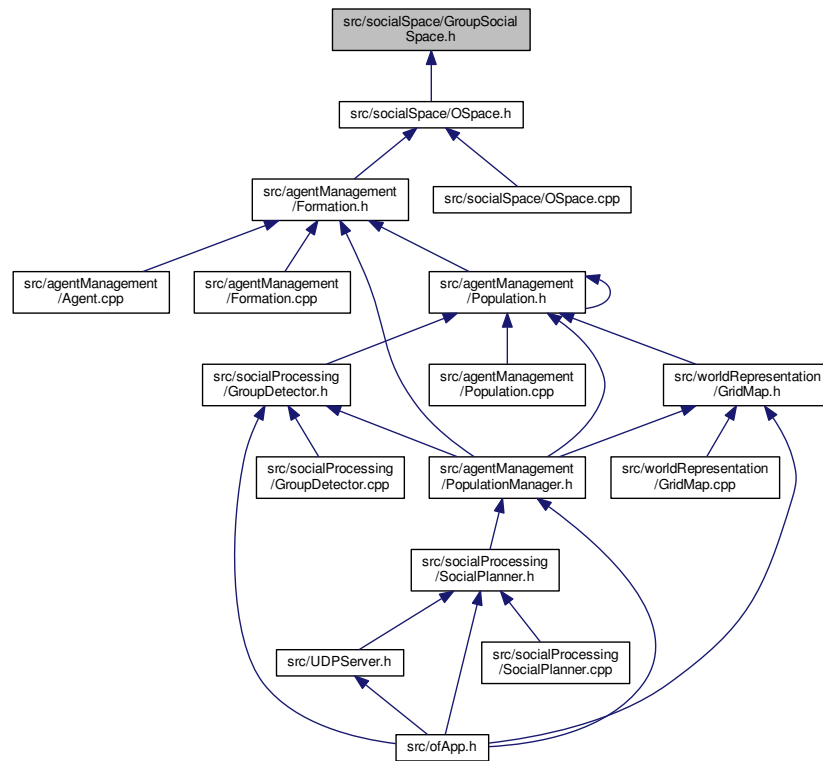
```
#include "SocialSpace.h"
```

```
#include "AgentContainer.h"
```

Include dependency graph for GroupSocialSpace.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [GroupSocialSpace](#)

This class is an interface to implement representation of a [GroupSocialSpace](#).

6.22.1 Detailed Description

Author

Paco Dupont

Version

0.1

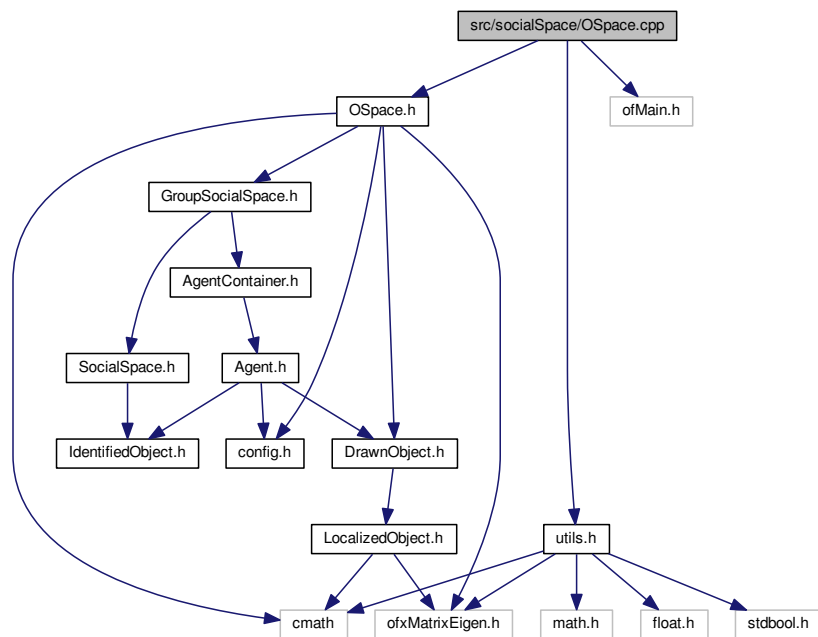
Date

29 mars 2017

6.23 src/socialSpace/OSpace.cpp File Reference

```
#include "OSpace.h"  
#include "utils.h"  
#include "ofMain.h"
```

Include dependency graph for OSpace.cpp:



6.23.1 Detailed Description

Author

Paco Dupont

Version

0.1

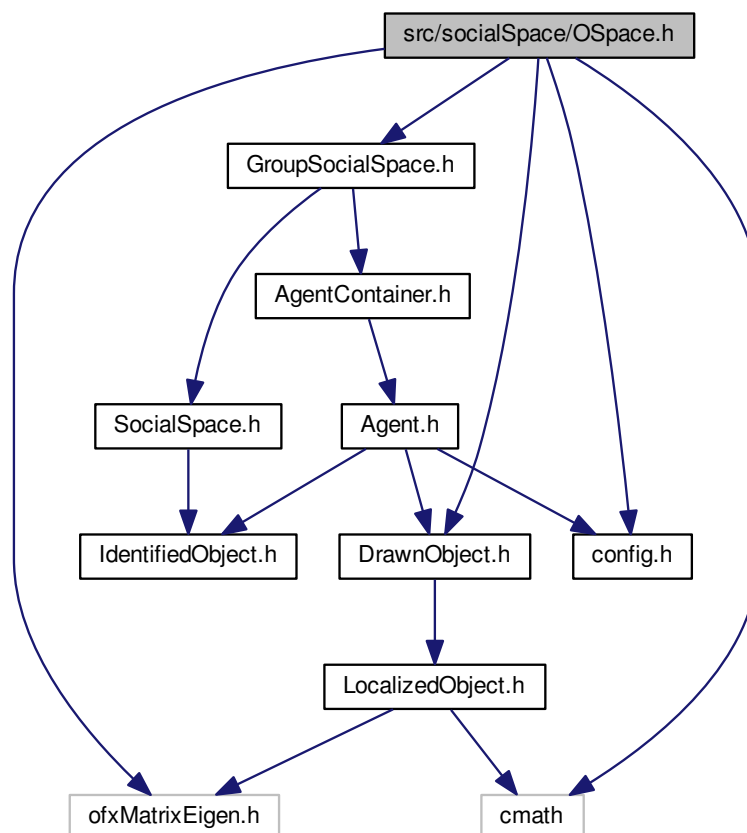
Date

27 mars 2017

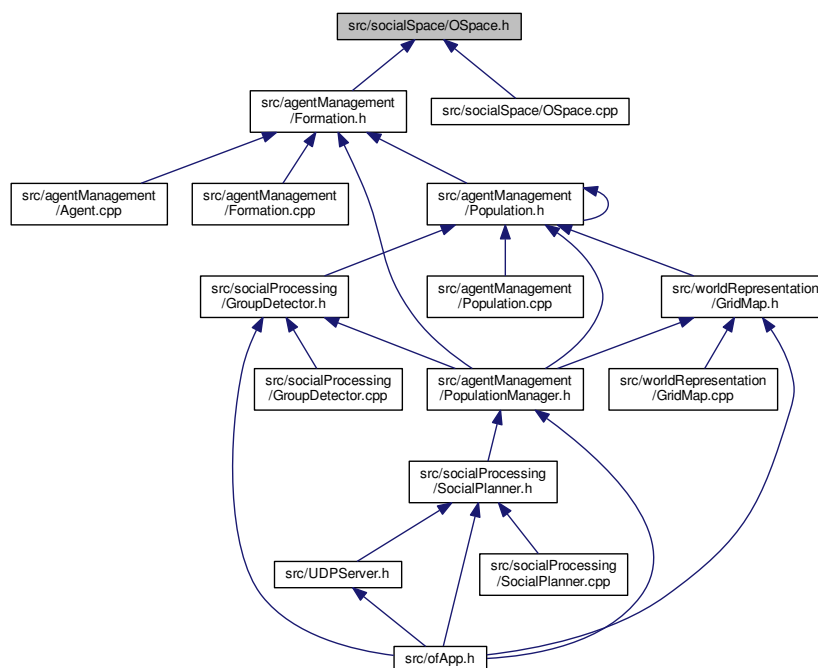
6.24 src/socialSpace/OSpace.h File Reference

```
#include "GroupSocialSpace.h"
#include "DrawnObject.h"
#include "config.h"
#include "ofxMatrixEigen.h"
#include <cmath>
```

Include dependency graph for OSpace.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OSpace

This class is an implementation of the [GroupSocialSpace](#).

6.24.1 Detailed Description

Author

Paco Dupont

Version

0.1

Date _____

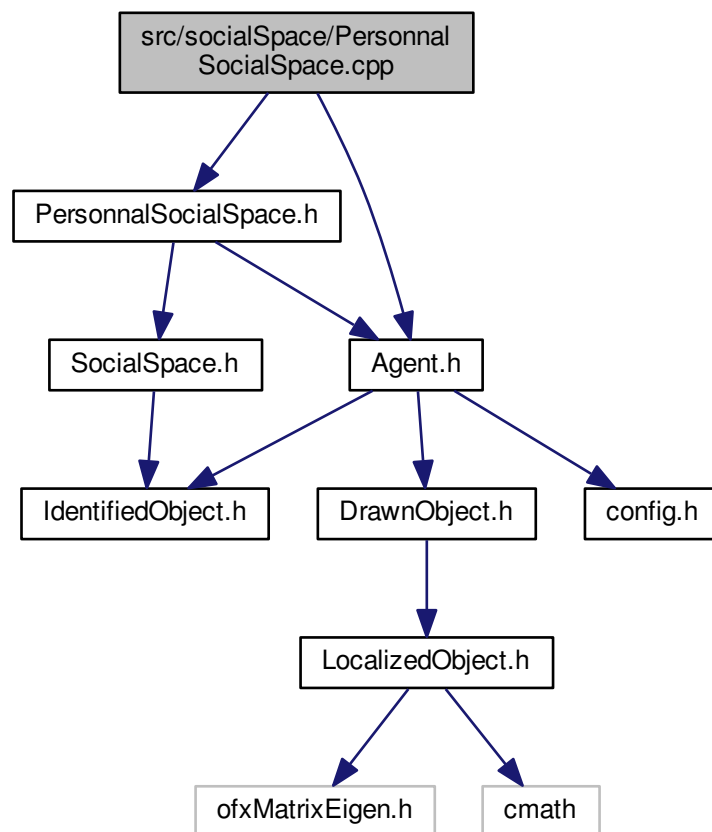
27 mars 2017

6.25 src/socialSpace/PersonnalSocialSpace.cpp File Reference

```
#include "PersonnalSocialSpace.h"
```

```
#include "Agent.h"
```

Include dependency graph for PersonnalSocialSpace.cpp:



6.25.1 Detailed Description

Author

Paco Dupont

Version

0.1

Date

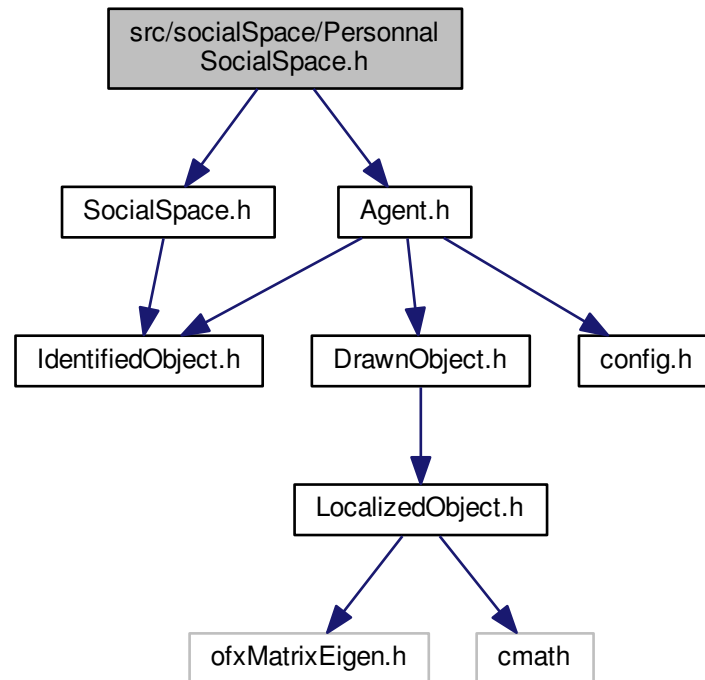
29 mars 2017

6.26 src/socialSpace/PersonnalSocialSpace.h File Reference

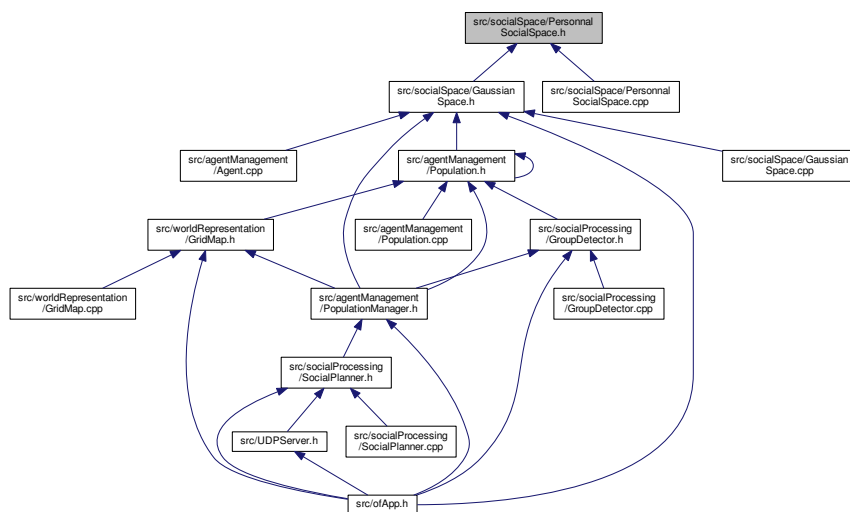
```
#include "SocialSpace.h"
```

```
#include "Agent.h"
```

Include dependency graph for PersonnalSocialSpace.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [PersonnalSocialSpace](#)

This class is an interface to implement representation of a [PersonnalSocialSpace](#).

6.26.1 Detailed Description

Author

Paco Dupont

Version

0.1

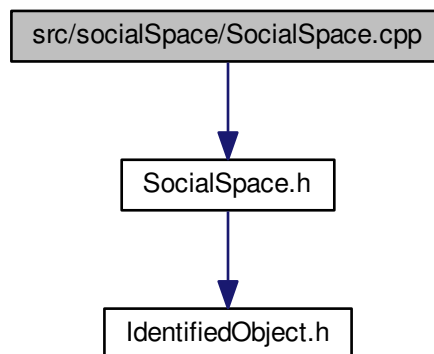
Date

29 mars 2017

6.27 src/socialSpace/SocialSpace.cpp File Reference

```
#include "SocialSpace.h"
```

Include dependency graph for SocialSpace.cpp:



6.27.1 Detailed Description

Author

Paco Dupont

Version

0.1

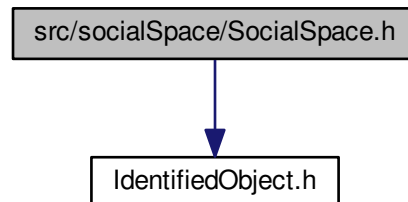
Date

27 mars 2017

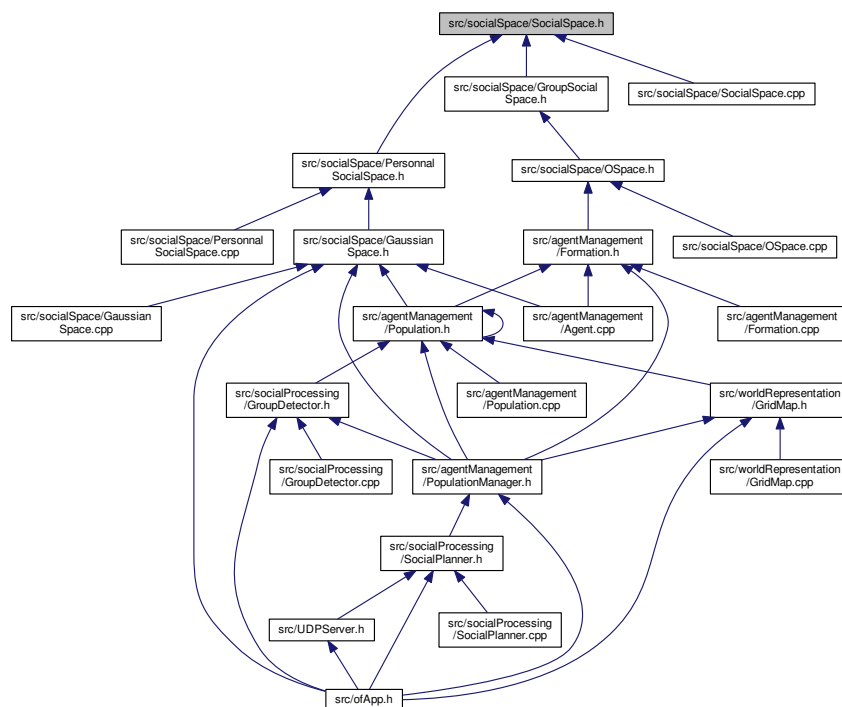
6.28 src/socialSpace/SocialSpace.h File Reference

```
#include "IdentifiedObject.h"
```

Include dependency graph for SocialSpace.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SocialSpace](#)

This class is an abstract class for representing [SocialSpace](#).

Classes

- class [UDPServer](#)

This class manage the UDP Server sending computed data to a visualization software and receiving [Agent](#) data from other sensor sources.

6.29.1 Detailed Description

Author

Paco Dupont

Version

0.1

Date

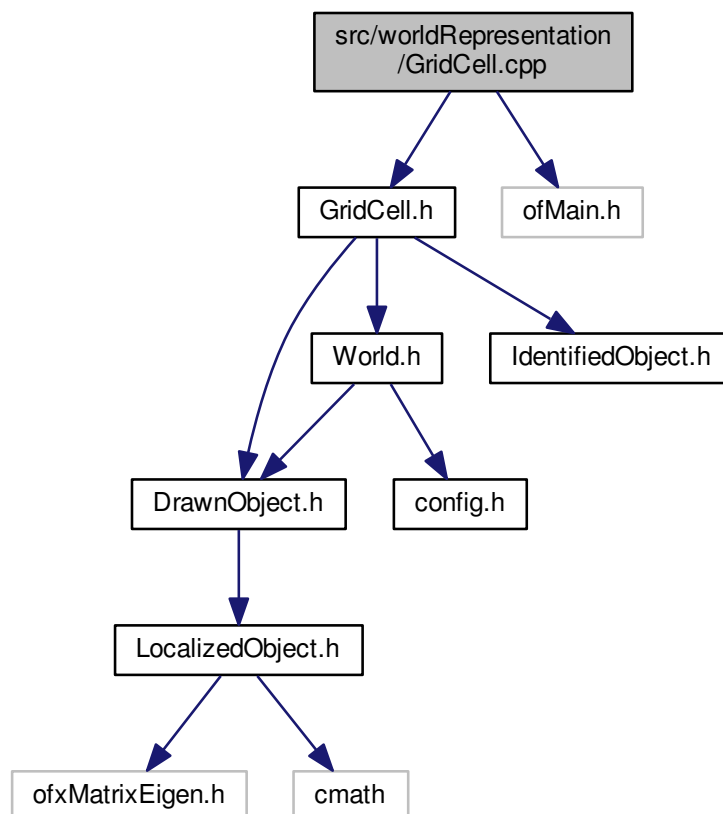
19 avril 2017

6.30 src/worldRepresentation/GridCell.cpp File Reference

```
#include "GridCell.h"
```

```
#include "ofMain.h"
```

Include dependency graph for GridCell.cpp:



6.30.1 Detailed Description

Author

Paco Dupont

Version

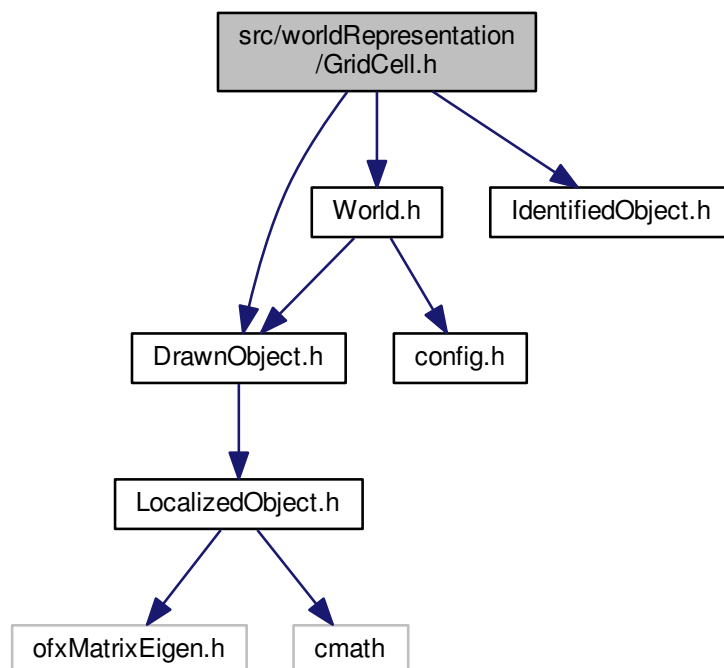
0.1

Date

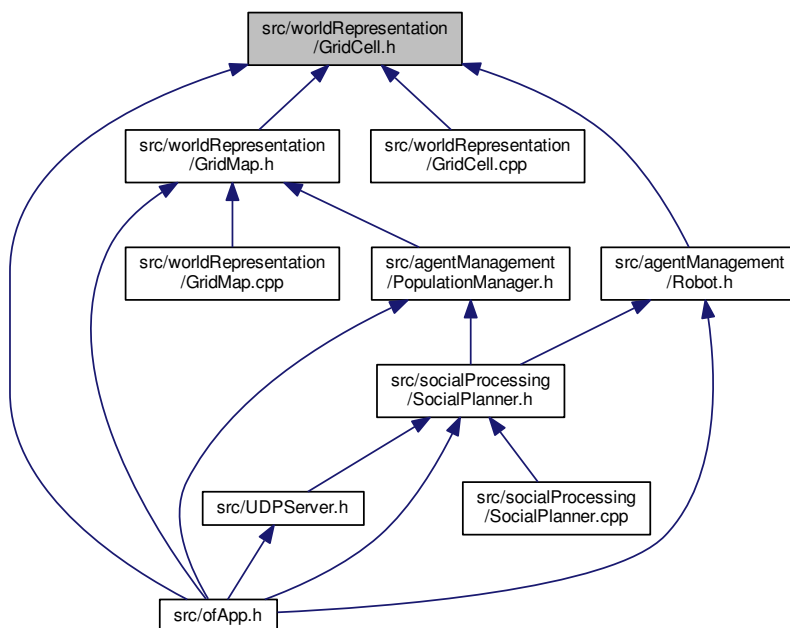
6 avril 2017

6.31 src/worldRepresentation/GridCell.h File Reference

```
#include "World.h"
#include "IdentifiedObject.h"
#include "DrawnObject.h"
Include dependency graph for GridCell.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [GridCell](#)

This class represent a cell in the [GridMap](#).

6.31.1 Detailed Description

Author

Paco Dupont

Version

0.1

Date

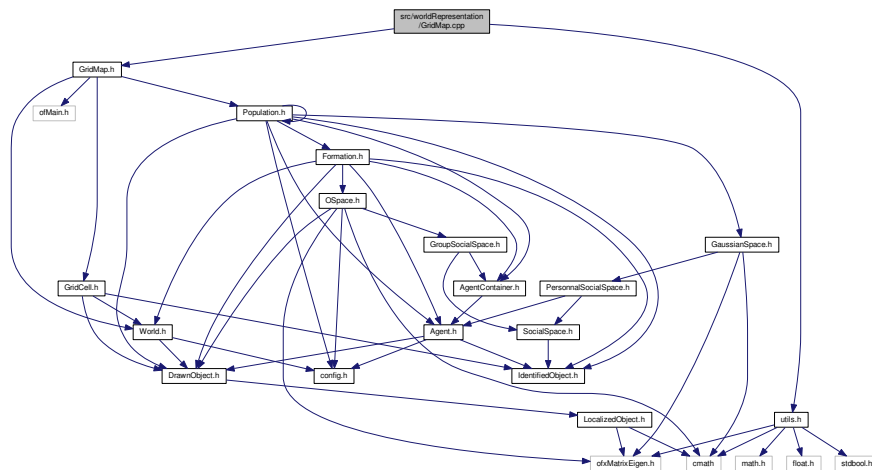
6 avril 2017

6.32 src/worldRepresentation/GridMap.cpp File Reference

```
#include "GridMap.h"
```

```
#include "utils.h"
```

Include dependency graph for GridMap.cpp:



Functions

- double [heuristicManhattanCostEstimate](#) ([GridCell](#) *start, [GridCell](#) *end)
Estimate the cost of movement from a [GridCell](#) to another.
- double [heuristicDiagonalCostEstimate](#) ([GridCell](#) *start, [GridCell](#) *end)
Estimate the cost of movement from a [GridCell](#) to another.

6.32.1 Detailed Description

Author

Paco Dupont

Version

0.1

Date

6 avril 2017

6.32.2 Function Documentation

6.32.2.1 double heuristicDiagonalCostEstimate ([GridCell](#) * start, [GridCell](#) * end)

Estimate the cost of movement from a [GridCell](#) to another.

Estimate the cost of movement from a [GridCell](#) to another with diagonal heuristic. This estimate is better for 8 movement allowed

Parameters

<i>start</i>	: The starting GridCell
<i>end</i>	: The target GridCell

Returns

The value of the estimated cost

6.32.2.2 double heuristicManhattanCostEstimate ([GridCell](#) * *start*, [GridCell](#) * *end*)

Estimate the cost of movement from a [GridCell](#) to another.

Estimate the cost of movement from a [GridCell](#) to another with manhattan heuristic. This estimate is better for 4 movement allowed

Parameters

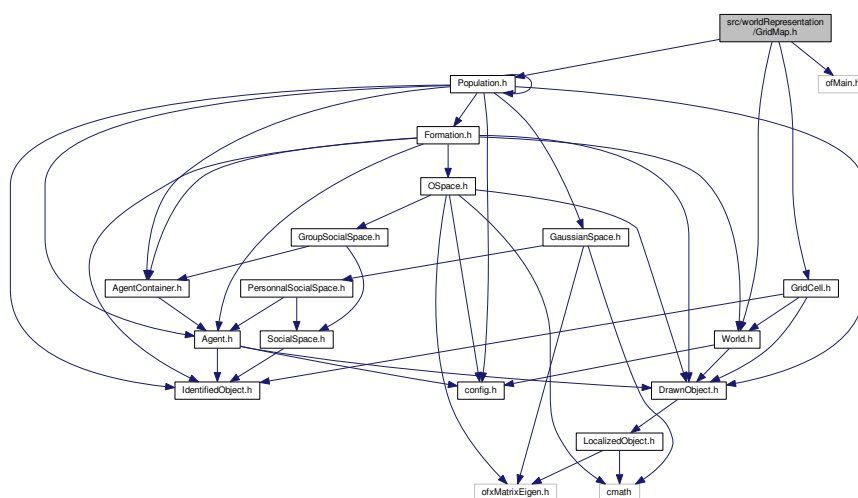
<i>start</i>	: The starting GridCell
<i>end</i>	: The target GridCell

Returns

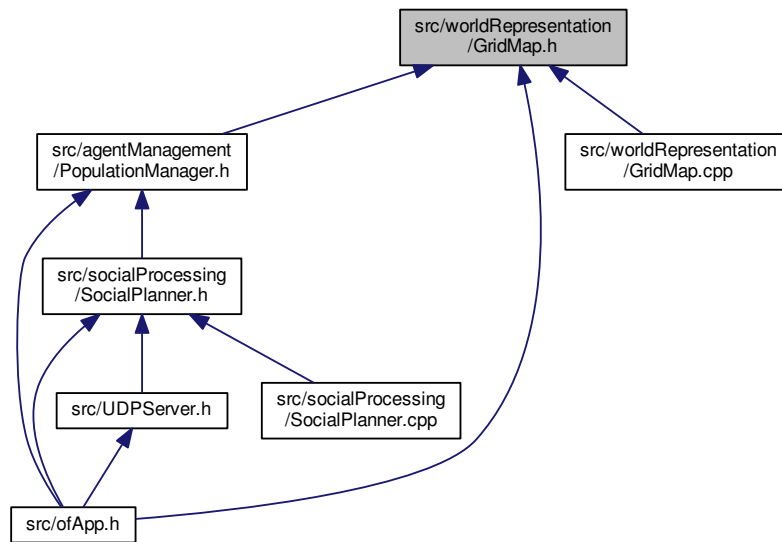
The value of the estimated cost

6.33 src/worldRepresentation/GridMap.h File Reference

```
#include "GridCell.h"
#include "World.h"
#include "Population.h"
#include "ofMain.h"
Include dependency graph for GridMap.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [GridMap](#)

This class manage the 2D [GridMap](#) computed from the Agents [SocialSpace](#).

- struct [GridMap::CompaireVCell](#)

Define an operator for [GridCell](#) and associated gScore comparison for A algorithm.*

6.33.1 Detailed Description

Author

Paco Dupont

Version

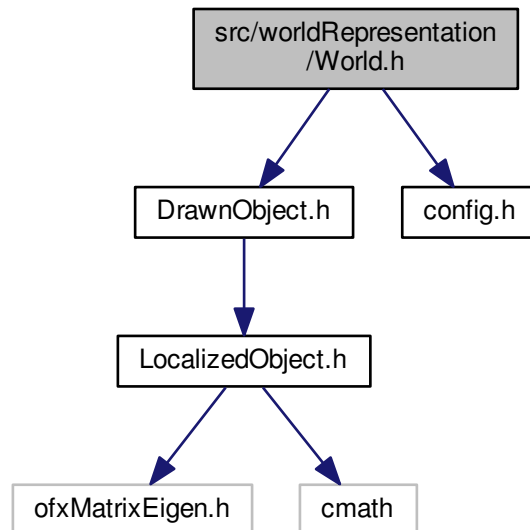
0.1

Date

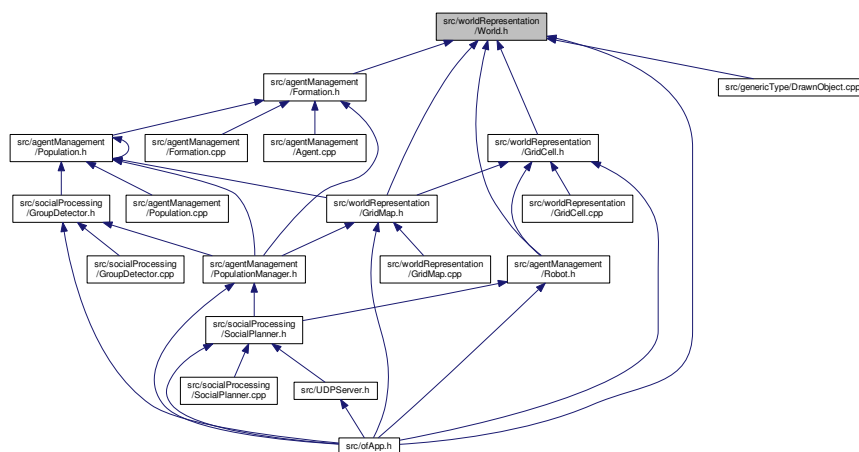
6 avril 2017

6.34 src/worldRepresentation/World.h File Reference

```
#include "DrawnObject.h"
#include "config.h"
Include dependency graph for World.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **World**

This class represent the main frame coordinates and its projection in pixels.

6.34.1 Detailed Description

Author

Paco Dupont

Version

0.1

Date

29 mars 2017

Index

- ~Agent
 - Agent, [11](#)
- ~AgentContainer
 - AgentContainer, [14](#)
- ~DrawnObject
 - DrawnObject, [17](#)
- ~Formation
 - Formation, [21](#)
- ~GaussianSpace
 - GaussianSpace, [26](#)
- ~GridCell
 - GridCell, [29](#)
- ~GridMap
 - GridMap, [37](#)
- ~GroupDetector
 - GroupDetector, [41](#)
- ~GroupSocialSpace
 - GroupSocialSpace, [43](#)
- ~IdentifiedObject
 - IdentifiedObject, [45](#)
- ~LocalizedObject
 - LocalizedObject, [47](#)
- ~OSpace
 - OSpace, [54](#)
- ~PersonnalSocialSpace
 - PersonnalSocialSpace, [58](#)
- ~Population
 - Population, [60](#)
- ~Robot
 - Robot, [68](#)
- ~SocialPlanner
 - SocialPlanner, [71](#)
- ~SocialSpace
 - SocialSpace, [73](#)
- ~UDPServer
 - UDPServer, [76](#)
- ~World
 - World, [80](#)
- Agent, [9](#)
 - ~Agent, [11](#)
 - Agent, [11](#)
 - findNearestNeighbor, [11](#)
 - getFOVIntersection, [11](#)
 - getSocialSpace, [12](#)
 - setSocialSpace, [12](#)
- AgentContainer, [12](#)
 - ~AgentContainer, [14](#)
 - AgentContainer, [14](#)
 - clearAgents, [14](#)
 - getAgent, [14](#)
 - getAgents, [14](#)
 - pushAgent, [15](#)
 - removeAgent, [15](#)
 - setAgents, [15](#)
- clear
 - Population, [60](#)
- clearAgents
 - AgentContainer, [14](#)
- clearFormations
 - Population, [60](#)
- computeCovarMatrix
 - OSpace, [54](#)
- constructPath
 - GridMap, [37](#)
- disengage
 - SocialPlanner, [71](#)
- do_read
 - UDPServer, [76](#)
- do_send
 - UDPServer, [76](#)
- DrawnObject, [16](#)
 - ~DrawnObject, [17](#)
 - DrawnObject, [17](#)
 - pixel_to_real, [18](#)
 - real_to_pixel, [18](#)
- engage
 - SocialPlanner, [71](#)
- findNearestNeighbor
 - Agent, [11](#)
- findPath
 - GridMap, [37](#)
- Formation, [18](#)
 - ~Formation, [21](#)
 - Formation, [21](#)
 - getInteractionDirection, [22](#)
 - getInteractionPosition, [22](#)
 - getInteractionPotential, [22](#)
 - getSocialSpace, [22](#)
 - initAgent, [22](#)
 - isInFormation, [23](#)
 - pushAgent, [23](#)
 - removeAgent, [23](#)
 - setInteractionDirection, [24](#)
 - setInteractionPosition, [24](#)
 - setInteractionPotential, [24](#)

- setSocialSpace, 24
- GaussianSpace, 24
 - ~GaussianSpace, 26
 - GaussianSpace, 26
 - phi, 26
- getAScore
 - GridCell, 30
- getAgent
 - AgentContainer, 14
- getAgents
 - AgentContainer, 14
- getCell
 - GridMap, 37
- getCenter
 - OSpace, 54
- getDirection
 - LocalizedObject, 47
- getFOVIntersection
 - Agent, 11
- getFormations
 - Population, 61
- getGoal
 - Robot, 68
- getHeight
 - World, 80
- getHeightView
 - World, 80
- getHighestFormationInteractionPotential
 - Population, 61
- getId
 - IdentifiedObject, 45
- getInteractionDirection
 - Formation, 22
- getInteractionPosition
 - Formation, 22
- getInteractionPotential
 - Formation, 22
- getManager
 - SocialPlanner, 71
- getPath
 - Robot, 68
- getPosition
 - LocalizedObject, 47
- getRelatedFormation
 - Population, 61
- getRobot
 - SocialPlanner, 71
- getSize
 - GridCell, 30
- getSocialSpace
 - Agent, 12
 - Formation, 22
- getTheta
 - LocalizedObject, 48
- getValue
 - GridCell, 30
- getWidth
 - World, 80
- getWidthView
 - World, 81
- getgCenter
 - OSpace, 54
- getX
 - LocalizedObject, 48
- getY
 - LocalizedObject, 48
- GridCell, 27
 - ~GridCell, 29
 - getAScore, 30
 - getSize, 30
 - getValue, 30
 - GridCell, 29
 - isBorderEnabled, 30
 - isCellSelected, 30
 - isFrontier, 30
 - isGoal, 31
 - isInfoEnabled, 31
 - isProcessed, 31
 - isStart, 31
 - setAScore, 31
 - setBorderEnabled, 32
 - setCellSelected, 32
 - setFrontier, 32
 - setGoal, 32
 - setInfoEnabled, 32
 - setProcessed, 33
 - setSize, 33
 - setStart, 33
 - setValue, 33
- GridMap, 33
 - ~GridMap, 37
 - constructPath, 37
 - findPath, 37
 - getCell, 37
 - GridMap, 36
 - isBorderEnabled, 38
 - isGroupSpaceEnabled, 38
 - isPersonalSpaceEnabled, 38
 - neighbors, 38
 - pathFinderNextStep, 39
 - setBorderEnabled, 39
 - setGroupSpaceEnabled, 39
 - setInfoEnabled, 39
 - setPersonalSpaceEnabled, 39
- GridMap.cpp
 - heuristicDiagonalCostEstimate, 120
 - heuristicManhattanCostEstimate, 121
- GridMap::CompaireVCell, 15
 - operator(), 16
- GroupDetector, 40
 - ~GroupDetector, 41
 - GroupDetector, 41
- GroupSocialSpace, 41
 - ~GroupSocialSpace, 43
 - GroupSocialSpace, 43
- Gui, 43

- heuristicDiagonalCostEstimate
 - GridMap.cpp, 120
- heuristicManhattanCostEstimate
 - GridMap.cpp, 121
- IdentifiedObject, 44
 - ~IdentifiedObject, 45
 - getId, 45
 - IdentifiedObject, 45
 - setId, 45
- initAgent
 - Formation, 22
- interact
 - SocialPlanner, 71
- isBorderEnabled
 - GridCell, 30
 - GridMap, 38
- isCellSelected
 - GridCell, 30
- isFrontier
 - GridCell, 30
- isGoal
 - GridCell, 31
- isGroupSpaceEnabled
 - GridMap, 38
- isGrouped
 - Population, 61, 62
- isInFormation
 - Formation, 23
- isInfoEnabled
 - GridCell, 31
- isPersonalSpaceEnabled
 - GridMap, 38
- isProcessed
 - GridCell, 31
- isStart
 - GridCell, 31
- less
 - OSpace, 54
- LocalizedObject, 46
 - ~LocalizedObject, 47
 - getDirection, 47
 - getPosition, 47
 - getTheta, 48
 - getX, 48
 - getY, 48
 - LocalizedObject, 47
 - setPosition, 48
 - setTheta, 48
 - setX, 49
 - setY, 49
- neighbors
 - GridMap, 38
- OSpace, 51
 - ~OSpace, 54
 - computeCovarMatrix, 54
 - getCenter, 54
 - getgCenter, 54
 - less, 54
 - OSpace, 53
 - phi, 55
 - setCenter, 55
 - setgCenter, 55
- ofApp, 49
- operator()
 - GridMap::CompaireVCell, 16
- parse
 - UDPServer, 76
- parse_frame0
 - UDPServer, 76
- pathFinderNextStep
 - GridMap, 39
- PersonnalSocialSpace, 56
 - ~PersonnalSocialSpace, 58
 - PersonnalSocialSpace, 57
- phi
 - GaussianSpace, 26
 - OSpace, 55
- pixel_to_real
 - DrawnObject, 18
- Population, 58
 - ~Population, 60
 - clear, 60
 - clearFormations, 60
 - getFormations, 61
 - getHighestFormationInteractionPotential, 61
 - getRelatedFormation, 61
 - isGrouped, 61, 62
 - Population, 60
 - pushFormation, 62
 - removeFormation, 62
 - setFormations, 62
- PopulationManager, 63
- pushAgent
 - AgentContainer, 15
 - Formation, 23
- pushFormation
 - Population, 62
- real_to_pixel
 - DrawnObject, 18
- removeAgent
 - AgentContainer, 15
 - Formation, 23
- removeFormation
 - Population, 62
- Robot, 65
 - ~Robot, 68
 - getGoal, 68
 - getPath, 68
 - Robot, 67
 - setGoal, 68
 - setPath, 68

- seek_interaction
 - SocialPlanner, 72
- send_frame0
 - UDPServer, 77
- send_frame1
 - UDPServer, 77
- send_frame2
 - UDPServer, 77
- send_frame3
 - UDPServer, 77
- setAStarScore
 - GridCell, 31
- setAgents
 - AgentContainer, 15
- setBorderEnabled
 - GridCell, 32
 - GridMap, 39
- setCellSelected
 - GridCell, 32
- setCenter
 - OSpace, 55
- setFormations
 - Population, 62
- setFrontier
 - GridCell, 32
- setGoal
 - GridCell, 32
 - Robot, 68
- setGroupSpaceEnabled
 - GridMap, 39
- setHeight
 - World, 81
- setHeightView
 - World, 81
- setId
 - IdentifiedObject, 45
- setInfoEnabled
 - GridCell, 32
 - GridMap, 39
- setInteractionDirection
 - Formation, 24
- setInteractionPosition
 - Formation, 24
- setInteractionPotential
 - Formation, 24
- setPath
 - Robot, 68
- setPersonalSpaceEnabled
 - GridMap, 39
- setPosition
 - LocalizedObject, 48
- setProcessed
 - GridCell, 33
- setSize
 - GridCell, 33
- setSocialSpace
 - Agent, 12
 - Formation, 24
- setStart
 - GridCell, 33
- setTheta
 - LocalizedObject, 48
- setValue
 - GridCell, 33
- setWidth
 - World, 81
- setWidthView
 - World, 81
- setgCenter
 - OSpace, 55
- setX
 - LocalizedObject, 49
- setY
 - LocalizedObject, 49
- SocialPlanner, 69
 - ~SocialPlanner, 71
 - disengage, 71
 - engage, 71
 - getManager, 71
 - getRobot, 71
 - interact, 71
 - seek_interaction, 72
 - SocialPlanner, 70
 - update, 72
- SocialSpace, 72
 - ~SocialSpace, 73
 - SocialSpace, 73
- spawn
 - UDPServer, 77
- src/UDPServer.h, 116
- src/agentManagement/Agent.cpp, 83
- src/agentManagement/Agent.h, 84
- src/agentManagement/Formation.cpp, 85
- src/agentManagement/Formation.h, 86
- src/agentManagement/Population.cpp, 88
- src/agentManagement/Population.h, 88
- src/agentManagement/Robot.h, 90
- src/genericType/AgentContainer.cpp, 92
- src/genericType/AgentContainer.h, 93
- src/genericType/DrawnObject.cpp, 95
- src/genericType/DrawnObject.h, 95
- src/genericType/IdentifiedObject.cpp, 97
- src/genericType/IdentifiedObject.h, 97
- src/genericType/LocalizedObject.cpp, 98
- src/genericType/LocalizedObject.h, 99
- src/socialProcessing/GroupDetector.cpp, 100
- src/socialProcessing/GroupDetector.h, 101
- src/socialProcessing/SocialPlanner.cpp, 102
- src/socialProcessing/SocialPlanner.h, 103
- src/socialSpace/GaussianSpace.cpp, 104
- src/socialSpace/GaussianSpace.h, 105
- src/socialSpace/GroupSocialSpace.h, 107
- src/socialSpace/OSpace.cpp, 109
- src/socialSpace/OSpace.h, 110
- src/socialSpace/PersonnalSocialSpace.cpp, 112
- src/socialSpace/PersonnalSocialSpace.h, 113

src/socialSpace/SocialSpace.cpp, [114](#)
src/socialSpace/SocialSpace.h, [115](#)
src/worldRepresentation/GridCell.cpp, [117](#)
src/worldRepresentation/GridCell.h, [118](#)
src/worldRepresentation/GridMap.cpp, [120](#)
src/worldRepresentation/GridMap.h, [121](#)
src/worldRepresentation/World.h, [123](#)

UDPServer, [74](#)

- ~UDPServer, [76](#)
- do_read, [76](#)
- do_send, [76](#)
- parse, [76](#)
- parse_frame0, [76](#)
- send_frame0, [77](#)
- send_frame1, [77](#)
- send_frame2, [77](#)
- send_frame3, [77](#)
- spawn, [77](#)
- UDPServer, [75](#)
- updateOrPushAgent, [77](#)

update

- SocialPlanner, [72](#)

updateOrPushAgent

- UDPServer, [77](#)

World, [78](#)

- ~World, [80](#)
- getHeight, [80](#)
- getHeightView, [80](#)
- getWidth, [80](#)
- getWidthView, [81](#)
- setHeight, [81](#)
- setHeightView, [81](#)
- setWidth, [81](#)
- setWidthView, [81](#)
- World, [80](#)