Social Planner

Generated by Doxygen 1.8.11

Contents

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:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

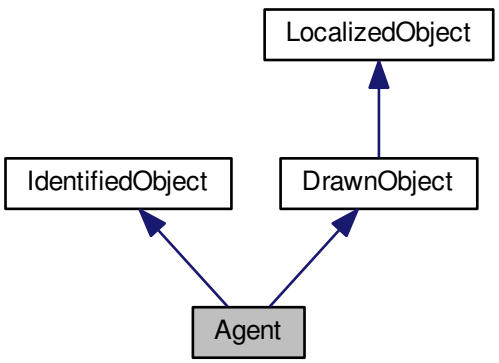
Class Documentation

5.1 Agent Class Reference

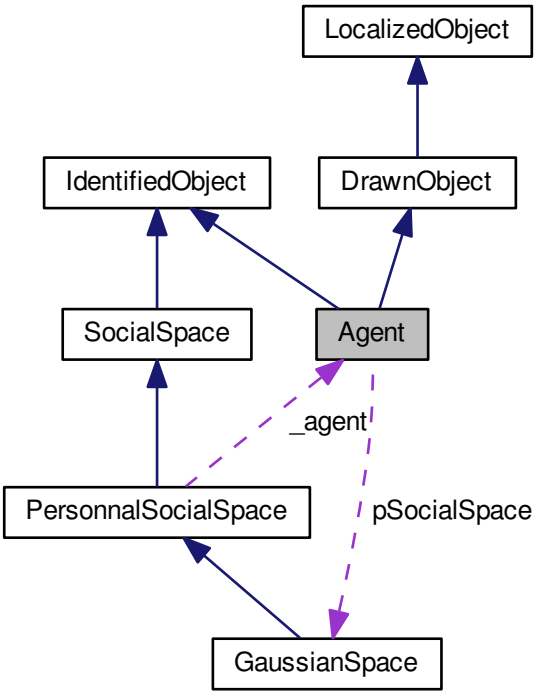
This class represent the Agents.

`#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

position	: Initial position of the Agent in World
theta	: Initial angle of the Agent
id	: 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

agents	: 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

agent	: 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

socialSpace	

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

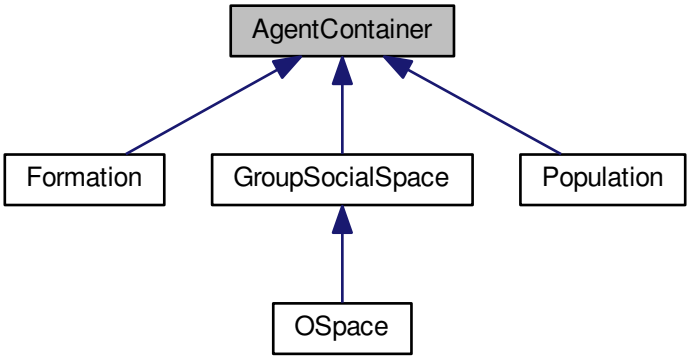
- src/agentManagement/Agent.h
- src/agentManagement/Agent.cpp

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

agents	: 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

agentId	: 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

agent	: The Agent to add

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

Remove an Agent by id from the container.

Parameters

agentId	: 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

agents	

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

a	: The VCell compared
b	: 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:

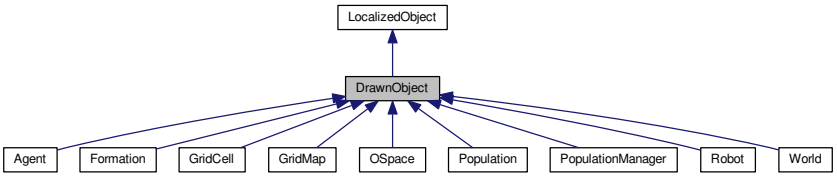
- src/worldRepresentation/GridMap.h

5.4 DrawnObject Class Reference

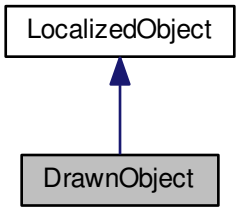
This class is an interface for class that are drawn on OFX gui.

`#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

position	: The initial position of the object
theta	: 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

world	: The World main coordinates frame
p	: 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

world	: The World main coordinates frame
p	: The vector that will be projected

Returns

The projected vector in pixel coordinates

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

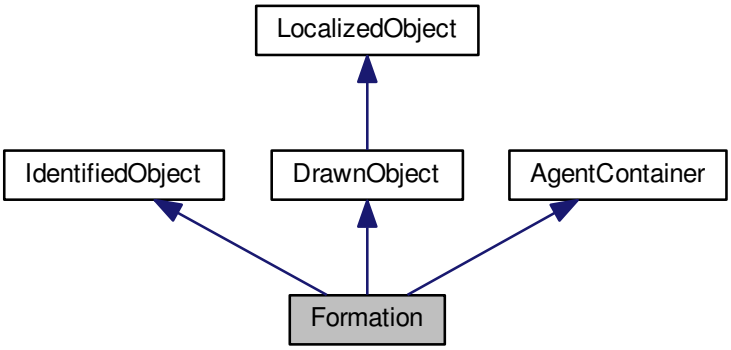
- src/genericType/DrawnObject.h
- src/genericType/DrawnObject.cpp

5.5 Formation Class Reference

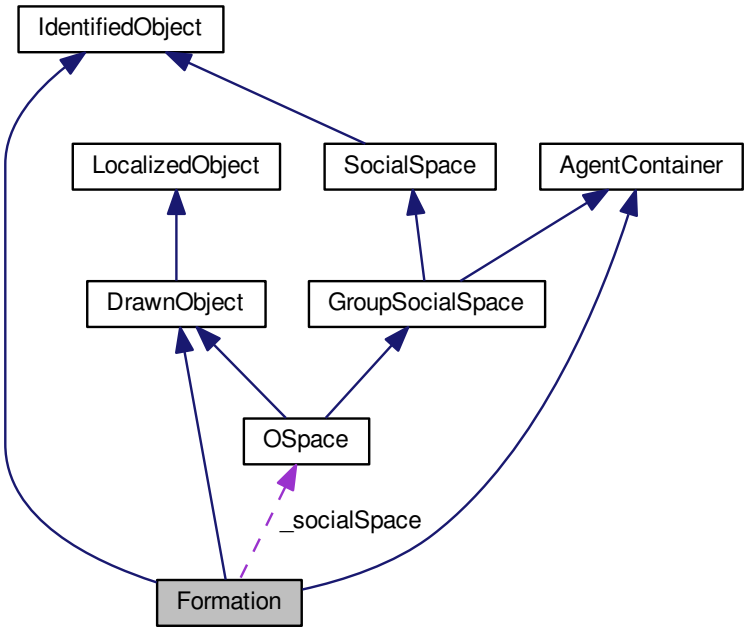
This class represent the social Formation.

`#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 find↩ InteractionPosition.
- 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

agents	: Agents that are part of the Formation
id	: 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

agent	: Agent that is part of the Formation
id	: 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 GroupSocialSpace

Parameters

id	: 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

agent	: 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

agent	: 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

agentId	: 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

agent	: 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

agentId	: Agent id to remove from the Formation

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

Simple setter.

Parameters

interactionDirection	

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

Simpler setter.

Parameters

interactionPosition	

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

Simple setter.

Parameters

interactionPotential	

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

Simple setter.

Parameters

socialSpace	

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

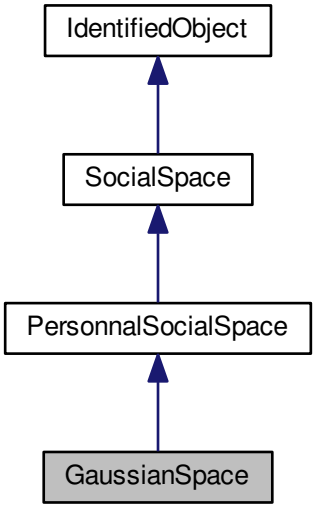
- src/agentManagement/Formation.h
- src/agentManagement/Formation.cpp

5.6 GaussianSpace Class Reference

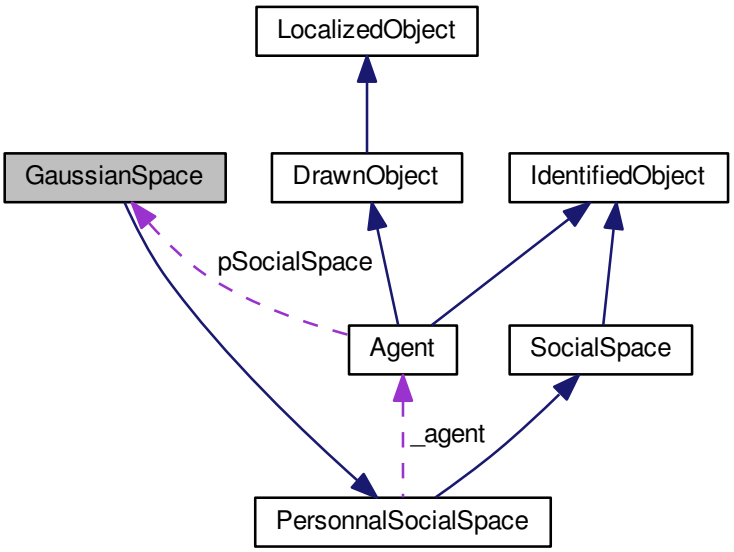
This class is an implementation of the PersonnalSocialSpace.

`#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

agent	: The Agent related to the GaussianSpace
id	: 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

testedRealPoint	: 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:

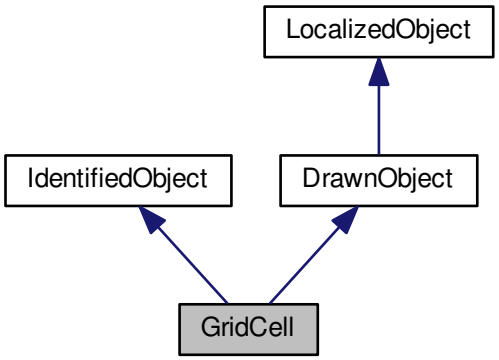
- src/socialSpace/GaussianSpace.h
- src/socialSpace/GaussianSpace.cpp

5.7 GridCell Class Reference

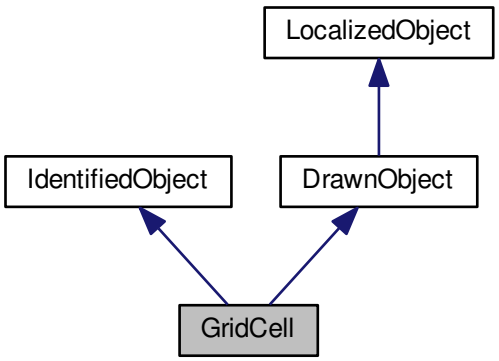
This class represent a cell in the GridMap.

`#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

size	: The size of the GridCell
value	: The initial value of the GridCell
position	: The position of the GridCell in real coordinate
id	: The unique identifier of the GridCell

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::getAStarScore ( ) 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::setAStarScore (int *starScore*)

Simple setter.

Parameters

starScore	

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

Simple setter.

Parameters

borderEnabled	

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

Simple setter.

Parameters

cellSelected	

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

Simple setter.

Parameters

frontier	

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

Simple setter.

Parameters

goal	

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

Simple setter.

Parameters

infoEnabled	

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

Simple setter.

Parameters

processed	

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

Simple setter.

Parameters

size	

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

Simple setter.

Parameters

start	

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

Simple setter.

Parameters

value	

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

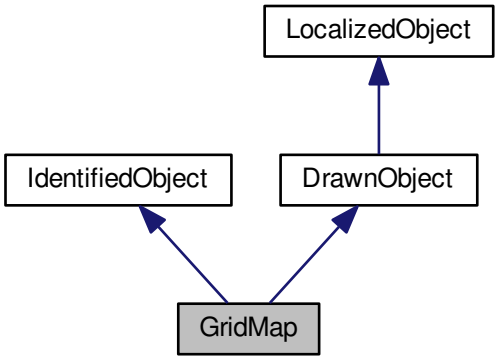
- src/worldRepresentation/GridCell.h
- src/worldRepresentation/GridCell.cpp

5.8 GridMap Class Reference

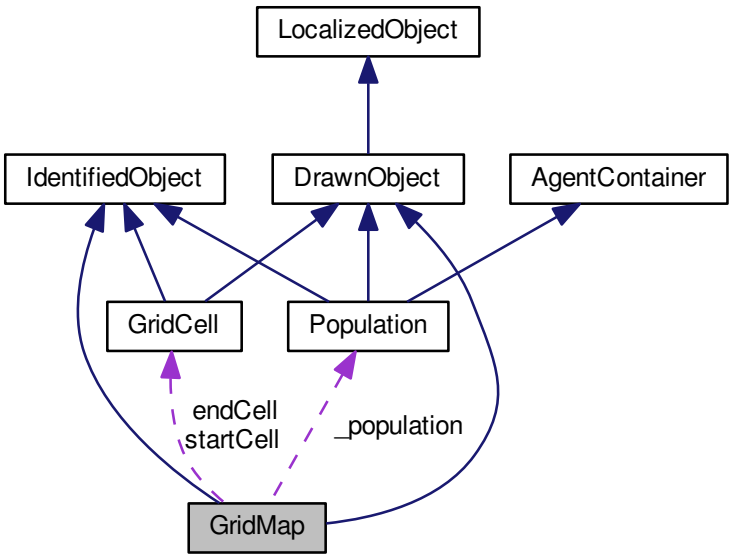
This class manage the 2D GridMap computed from the Agents SocialSpace.

`#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

world	: The main frame coordinates
pop	: The population related to this GridMap
resolution	: The resolution of the GridMap, must be a fraction of the World 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

startCell	: The starting GridCell
endCell	: 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

cellId	: 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

x	: The x coordinate of the GridCell in meter
y	: 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

cell	: Target GridCell
allowDiagonalMove	: 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

borderEnabled	

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

Simple setter.

Parameters

groupSpaceEnabled	

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

Simple setter.

Parameters

infoEnabled	

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

Simple setter.

Parameters

personalSpaceEnabled	

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

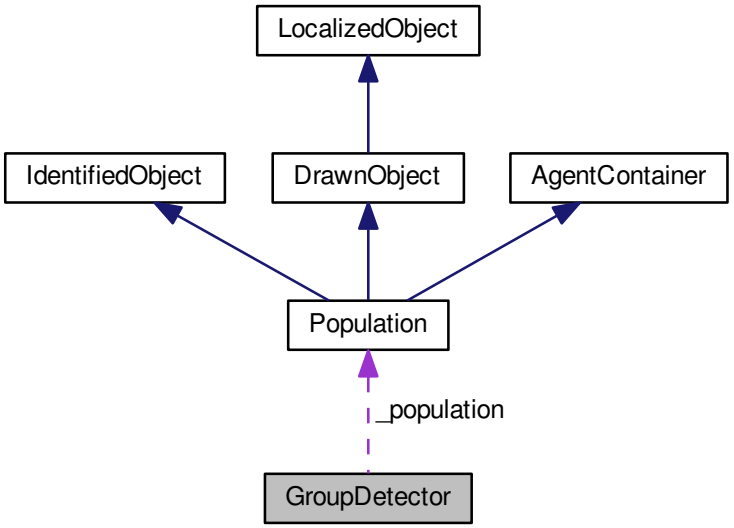
- src/worldRepresentation/GridMap.h
- src/worldRepresentation/GridMap.cpp

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

pop	: 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:

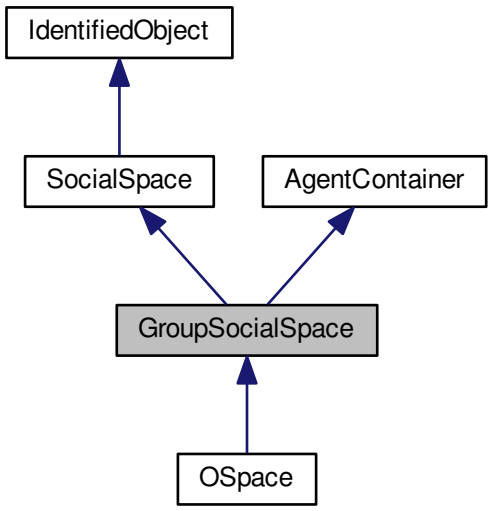
- src/socialProcessing/GroupDetector.h
- src/socialProcessing/GroupDetector.cpp

5.10 GroupSocialSpace Class Reference

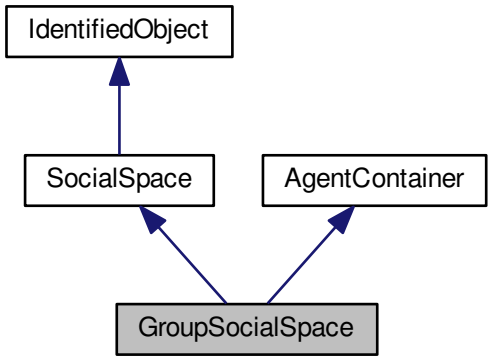
This class is an interface to implement representation of a GroupSocialSpace.

```
#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

id	: 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

agents	: The Agents related to the GroupSocialSpace
id	: 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:

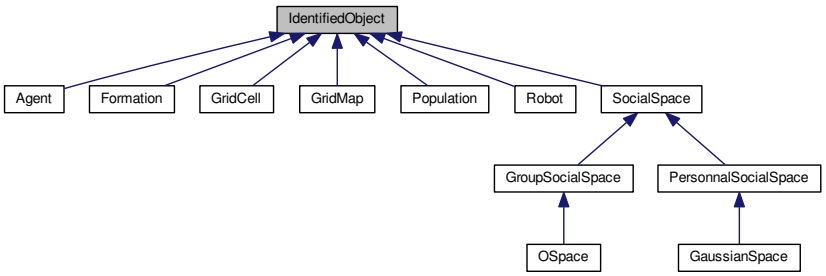
- src/Gui.h
- src/Gui.cpp

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

id	: 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

id	

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

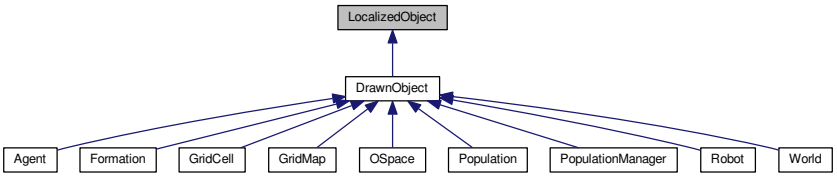
- src/genericType/IdentifiedObject.h
- src/genericType/IdentifiedObject.cpp

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

position	: The initial position of the object
theta	: 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

position	

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

Simple setter.

Parameters

theta	

5.13.3.8 void LocalizedObject::setX (double *x*)

Simple setter.

Parameters

x	

5.13.3.9 void LocalizedObject::setY (double *y*)

Simple setter.

Parameters

y	

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

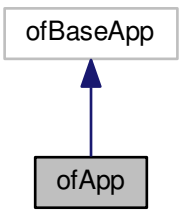
- src/genericType/LocalizedObject.h
- src/genericType/LocalizedObject.cpp

5.14 ofApp Class Reference

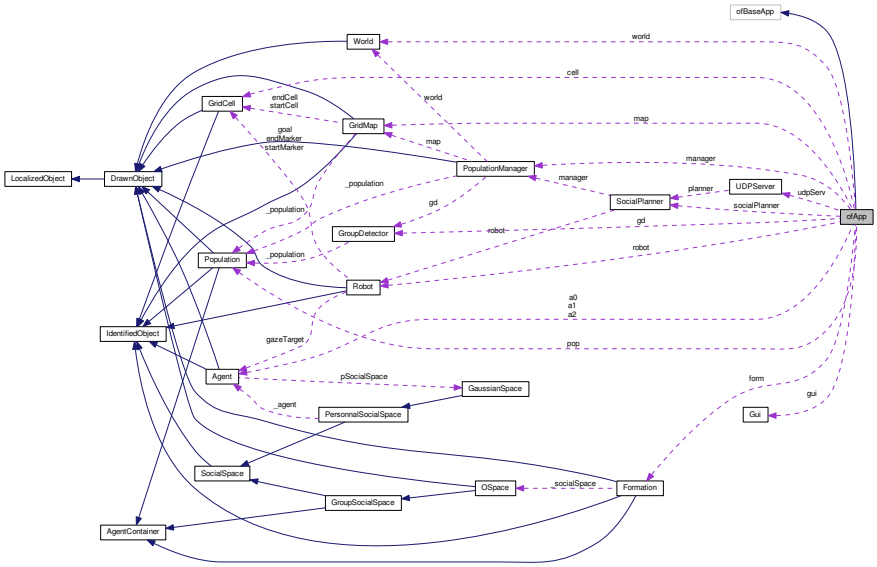
The Openframeworks main class.

`#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:

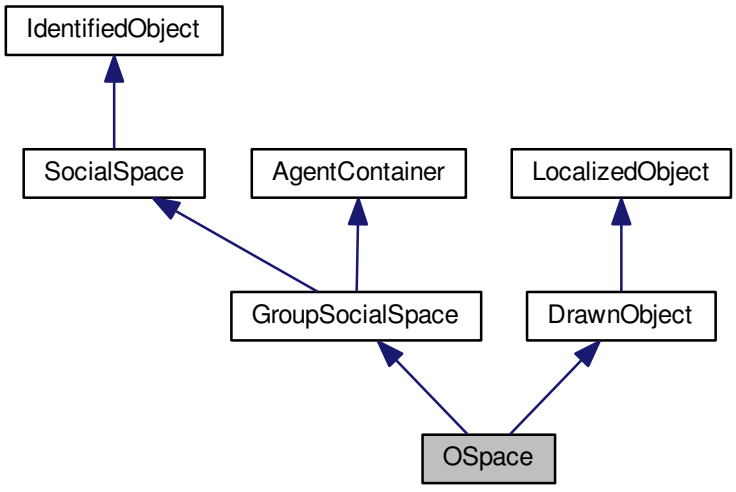
- src/ofApp.h
- src/ofApp.cpp

5.15 OSpace Class Reference

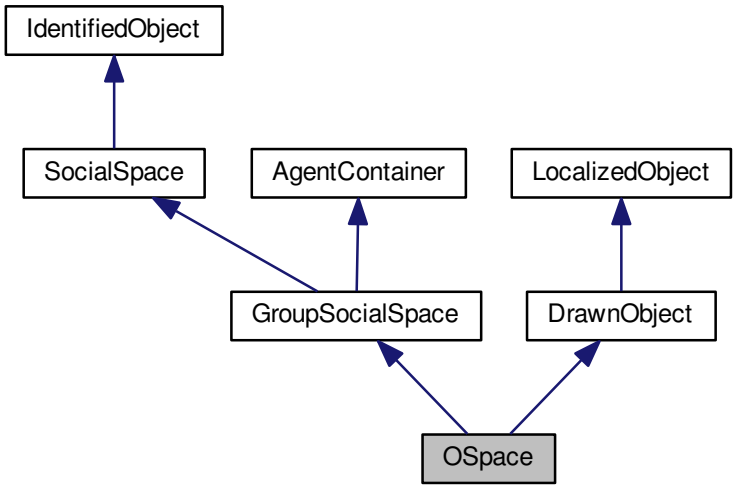
This class is an implementation of the GroupSocialSpace.

```
#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 computegCenter ()

  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 &center)

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

id	: 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

agents	: The Agents related to the GroupSocialSpace
id	: 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

a	: The first Agent position
b	: 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

testedPoint	: 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

center	

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

Simple setter.

Parameters

gCenter	

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

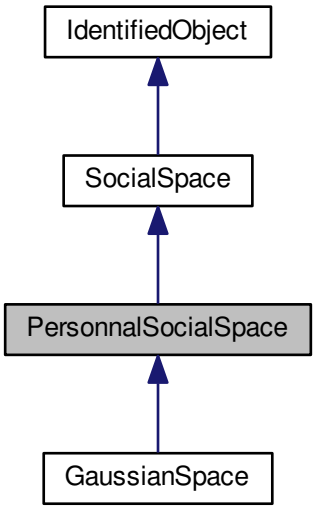
- src/socialSpace/OSpace.h
- src/socialSpace/OSpace.cpp

5.16 PersonnalSocialSpace Class Reference

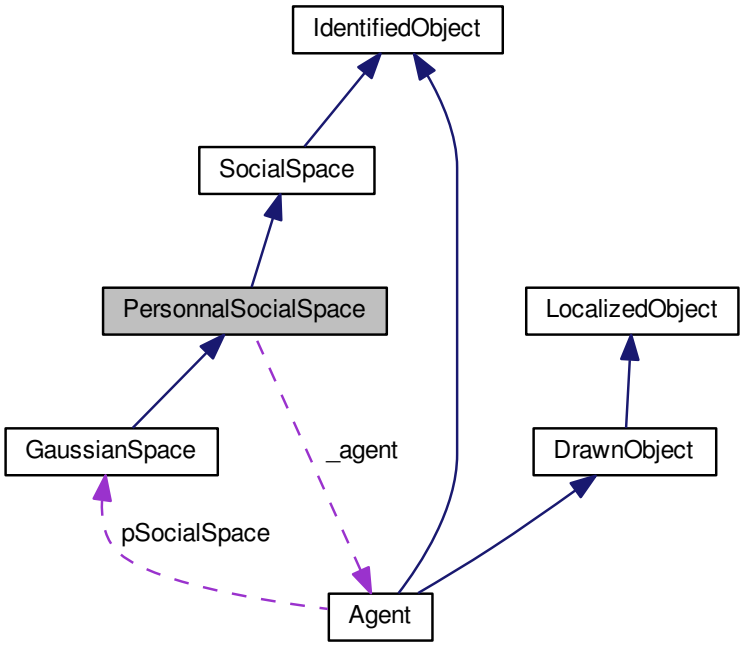
This class is an interface to implement representation of a PersonnalSocialSpace.

```
#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

agent	: The Agent related to the PersonnalSocialSpace
id	: 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:

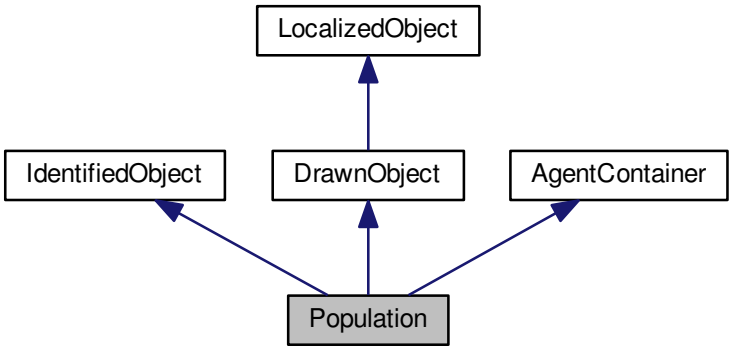
- src/socialSpace/PersonnalSocialSpace.h
- src/socialSpace/PersonnalSocialSpace.cpp

5.17 Population Class Reference

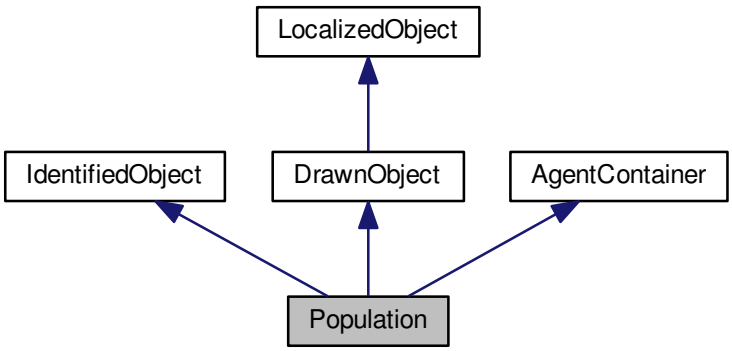
This class represent Population around the Robot.

`#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

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

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

Constructor.

Constructor of the Population class

Parameters

position	: Position of the Population (required by DrawnObject but useless here)
id	: 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

agent	: 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

agentId	: 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

formation	: 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

formationId	: 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

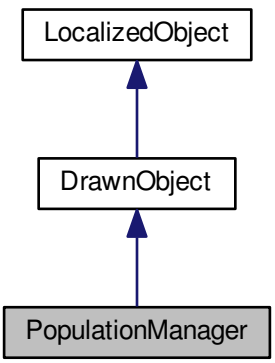
formations	

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

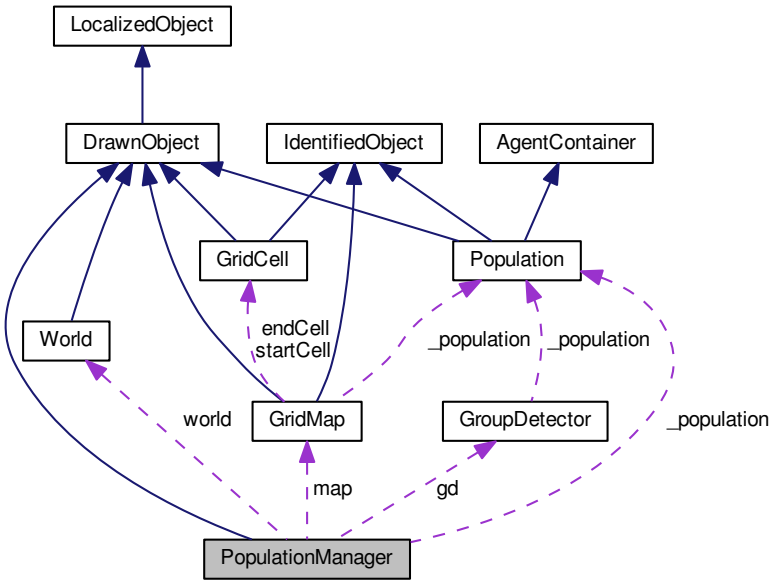
- src/agentManagement/Population.h
- src/agentManagement/Population.cpp

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 fIndex)
- 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 &gtFile)
- bool **isLoaded** () 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:

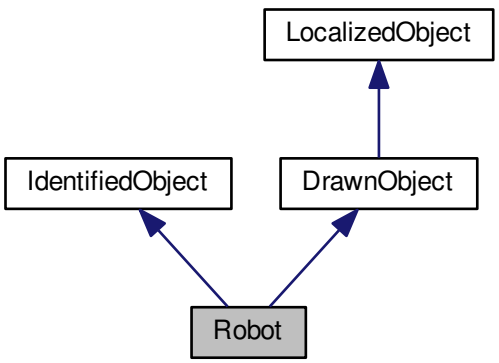
- src/agentManagement/PopulationManager.h
- src/agentManagement/PopulationManager.cpp

5.19 Robot Class Reference

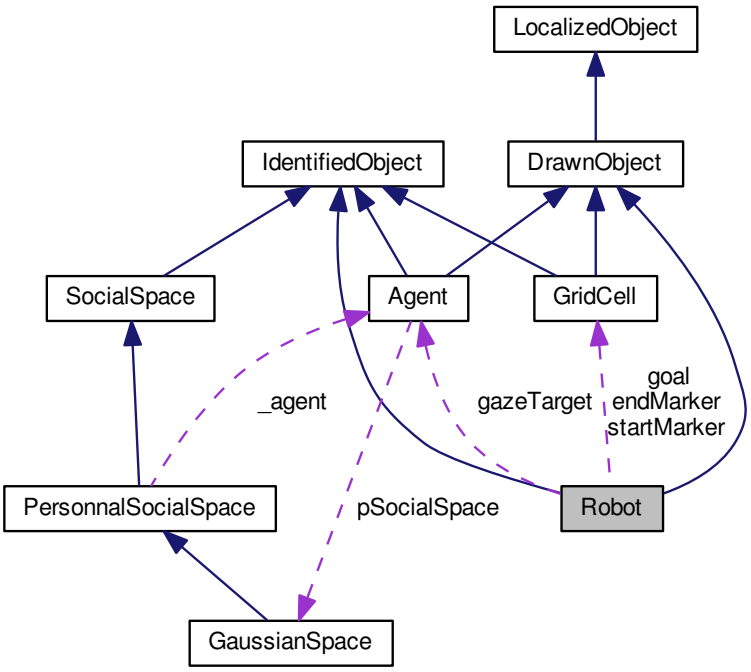
This class represent the Robot.

`#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

position	: Initial position of the Robot in World
theta	: 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

goal	

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

Simple setter.

Parameters

path	

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

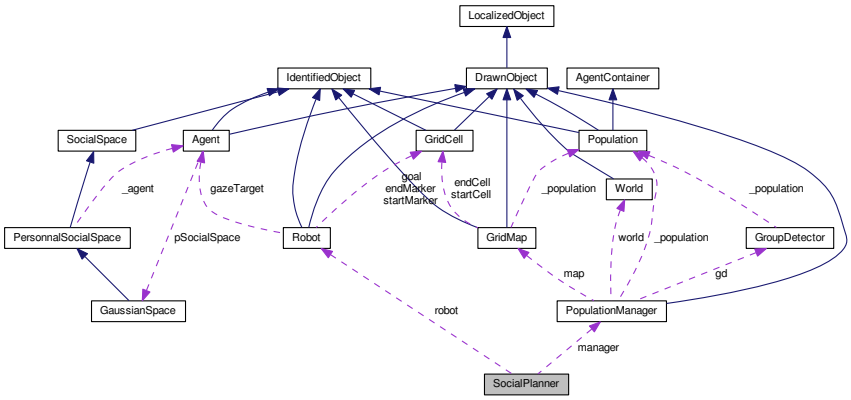
- src/agentManagement/Robot.h
- src/agentManagement/Robot.cpp

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

popManager	: The Population where the robot evolves
robot	: 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_POTEN↩ TIAL_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:

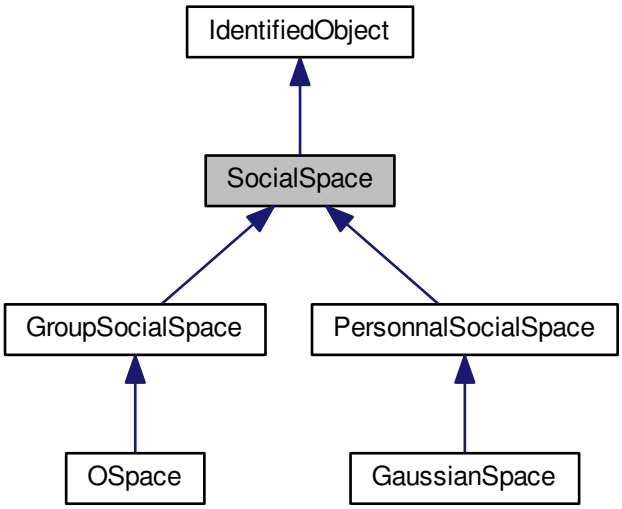
- src/socialProcessing/SocialPlanner.h
- src/socialProcessing/SocialPlanner.cpp

5.21 SocialSpace Class Reference

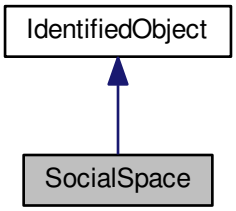
This class is an abstract class for representing SocialSpace.

`#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

id	: 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:

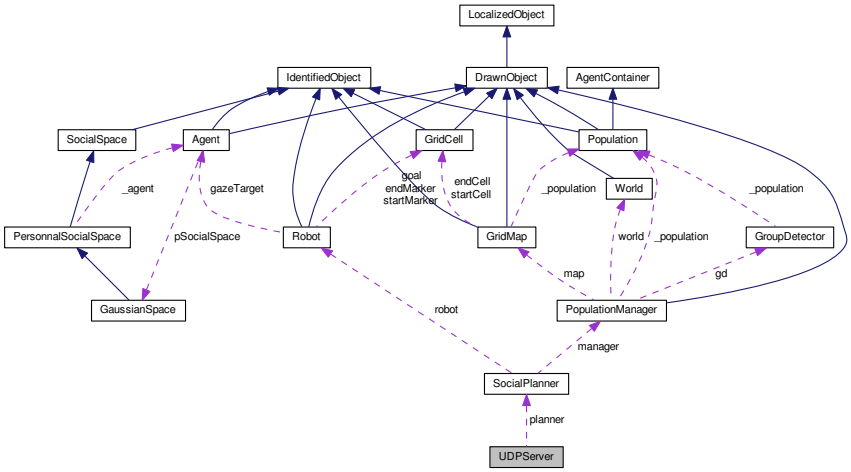
- src/socialSpace/SocialSpace.h
- src/socialSpace/SocialSpace.cpp

5.22 UDPServer Class Reference

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

`#include <UDPServer.h>`

Collaboration diagram for UDPServer:



Public Member Functions

- UDPServer (int port, SocialPlanner ∗sPlanner)

  Constructor.
- virtual ∼UDPServer ()

  Destructor.
- std::thread spawn ()

  Launch the UDPServer receive function in a new thread.
- int parse ()

  Parse the data received by the receive thread.
- int parse_frame0 ()

  Parse frame0 containing Agent data.
- int send_frame0 ()

  Create frame0 containing all Agent data from the Population.
- int send_frame1 ()

  Create frame1 containing the Robot data.
- int send_frame2 ()

  Create frame2 containing Robot path.
- int send_frame3 ()

  Create frame3 containing Formation data.

- 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

port	: The server port
sPlanner	: 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

sendBuffer	: The data buffer
sendBuffer_size	: 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

id	: The unique identifier of the Agent
x	: The x coordinate of the Agent
y	: The y coordinate of the Agent
z	: The z coordinate of the Agent
theta	: The angle of the Agent

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

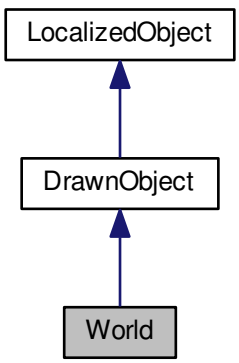
- src/UDPServer.h
- src/UDPServer.cpp

5.23 World Class Reference

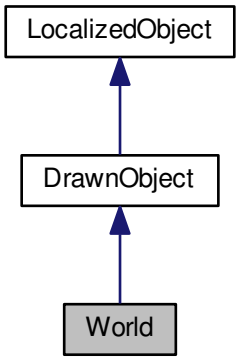
This class represent the main frame coordinates and its projection in pixels.

```
#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

width	: The real width of the World in meter
height	: The real height of the World in meter
widthView	: The projection width of the World in pixel
heightView	: The projection height of the World in pixel
position	: The position of the World in the GUI in pixel
theta	: 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

height	

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

Simple setter.

Parameters

heightView	

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

Simple setter.

Parameters

width	

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

Simple setter.

Parameters

widthView	

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

- src/worldRepresentation/World.h
- src/worldRepresentation/World.cpp

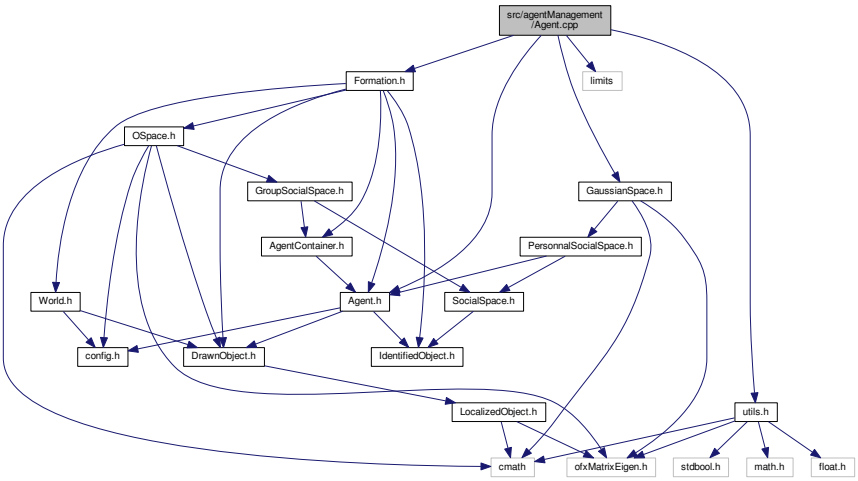
Chapter 6

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

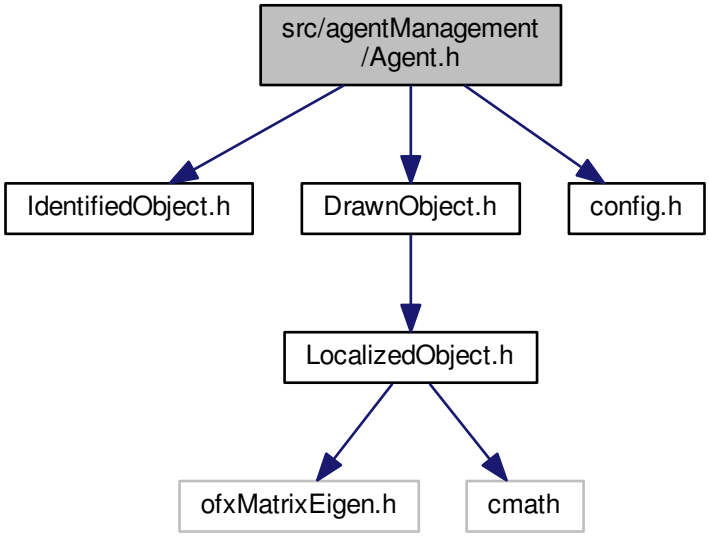
Date

24 mars 2017

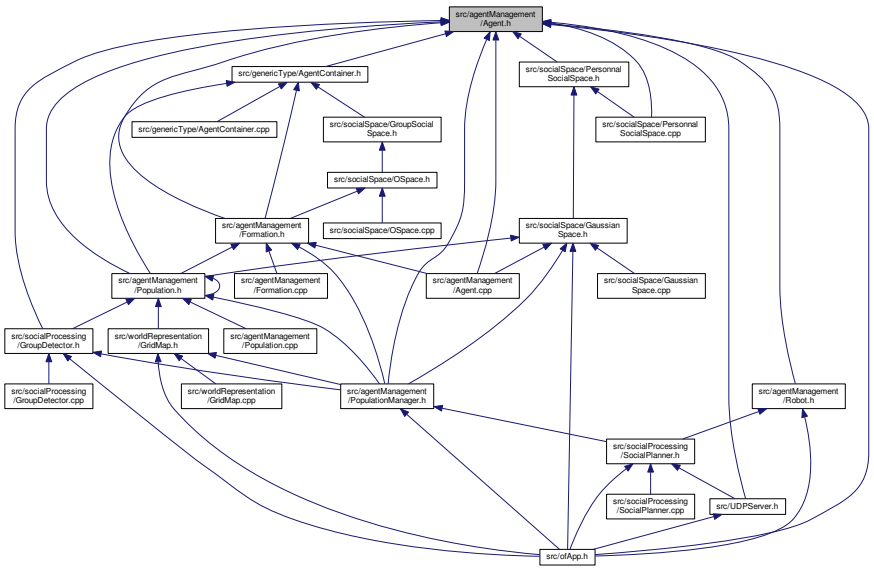
6.2 src/agentManagement/Agent.h File Reference

```
#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

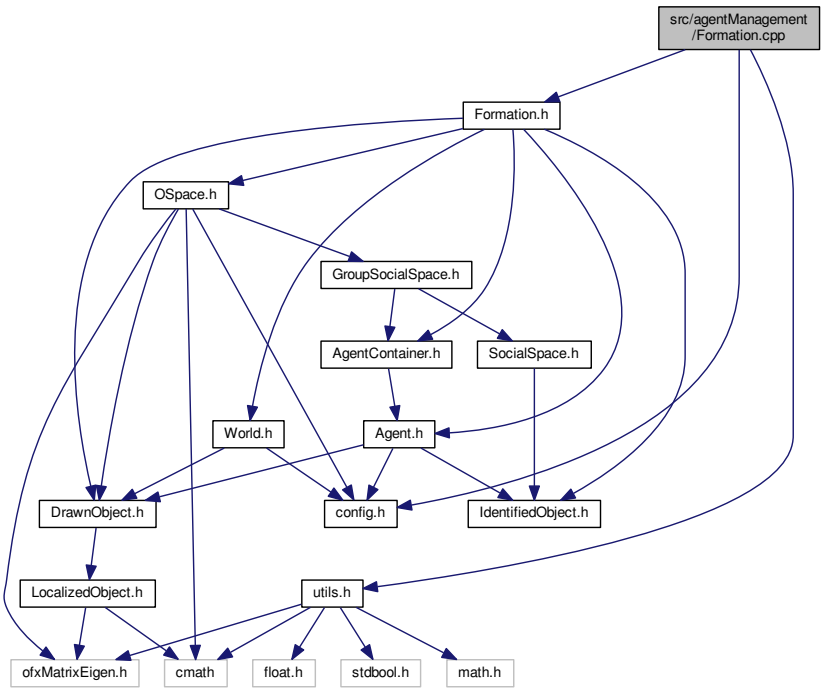
Date

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

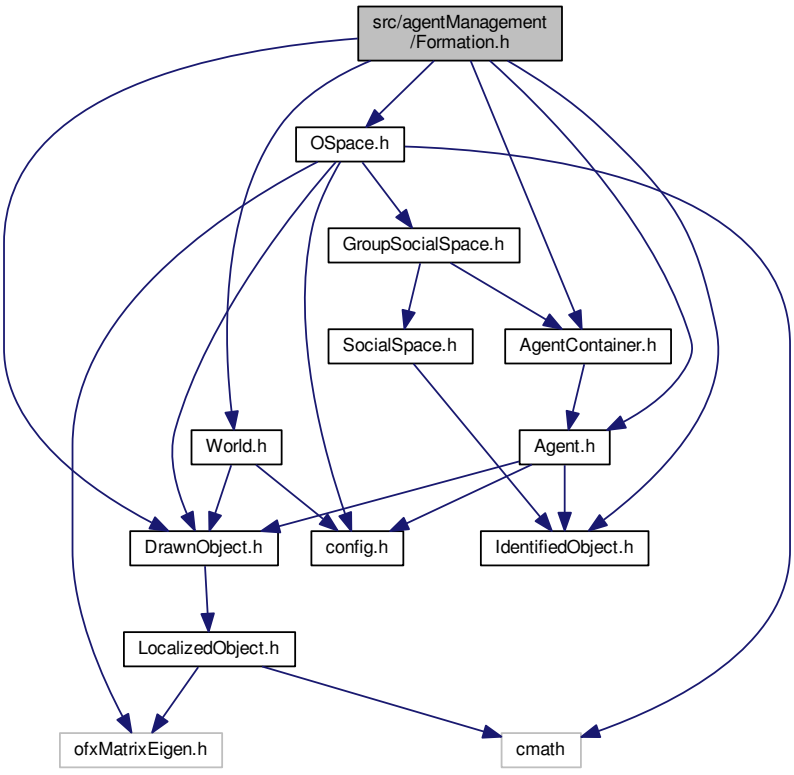
Date

24 mars 2017

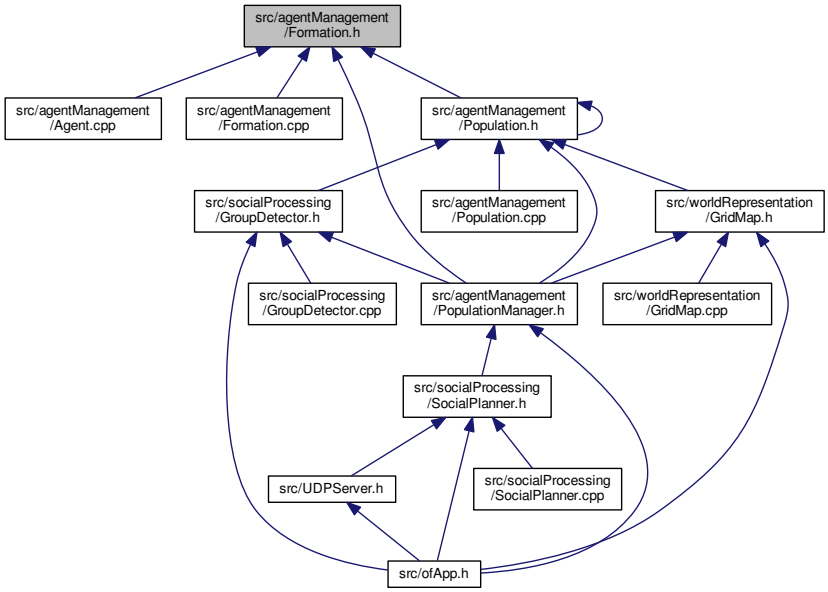
6.4 src/agentManagement/Formation.h File Reference

```
#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

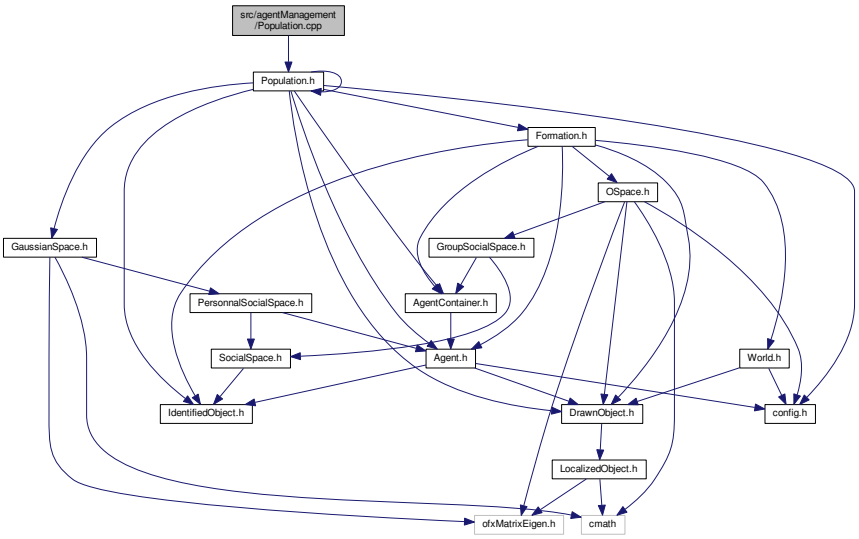
Date

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

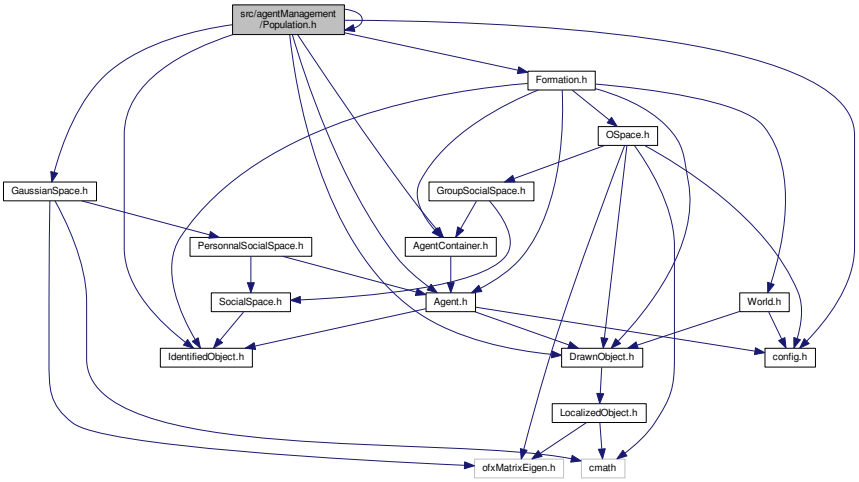
Date

24 mars 2017

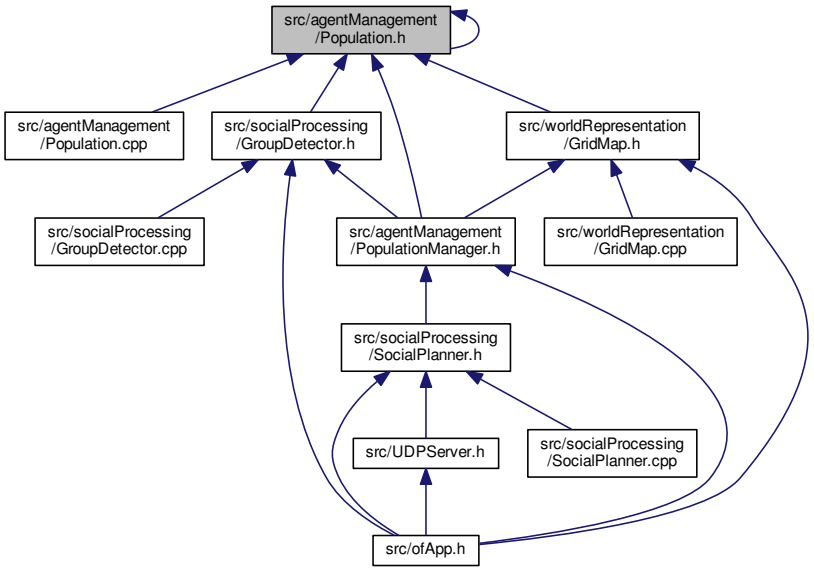
6.6 src/agentManagement/Population.h File Reference

```
#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

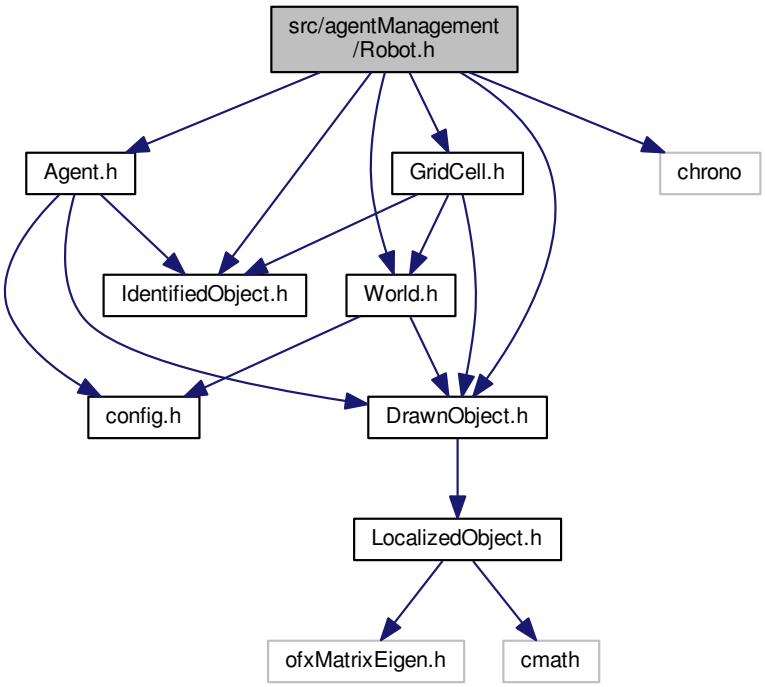
Date

24 mars 2017

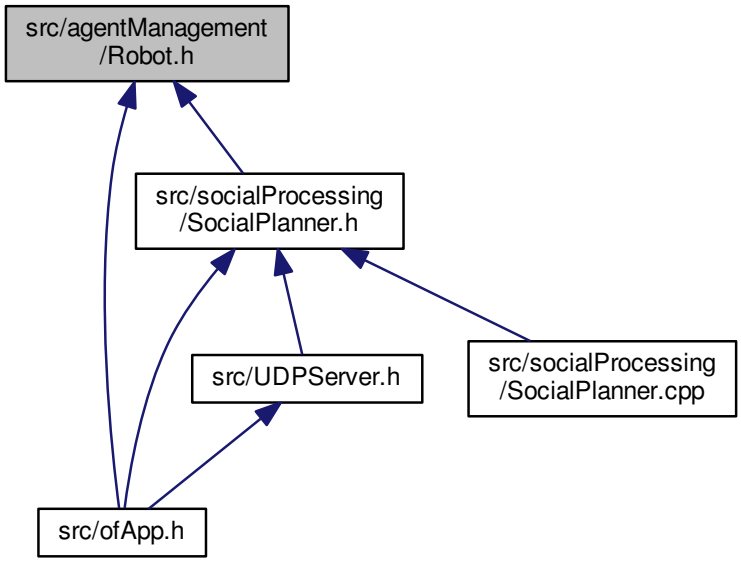
6.7 src/agentManagement/Robot.h File Reference

```
#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

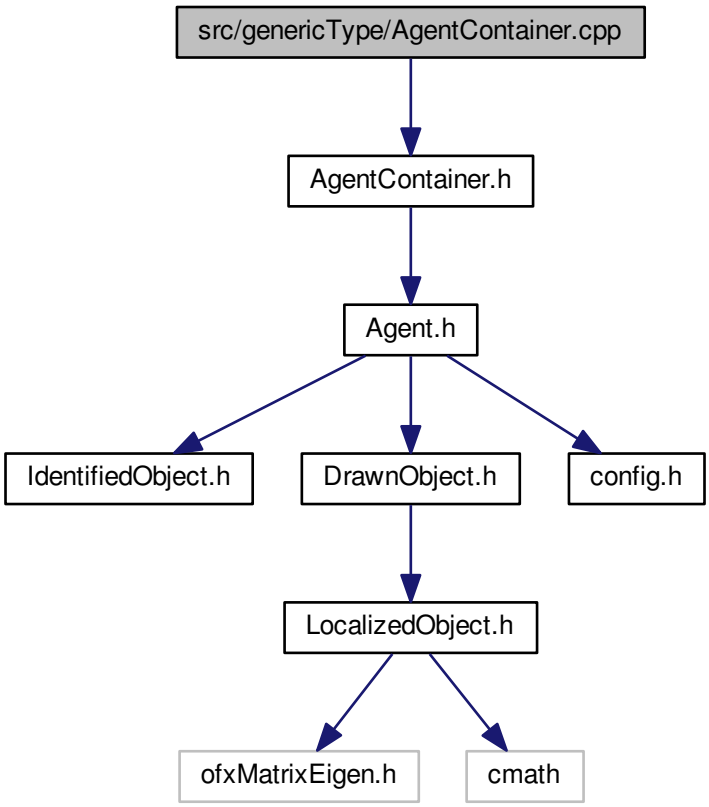
Date

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

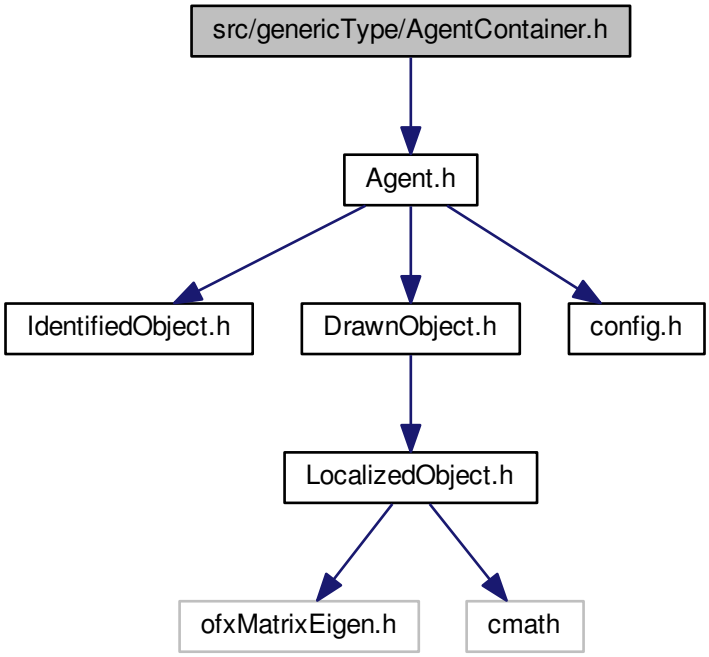
Date

27 mars 2017

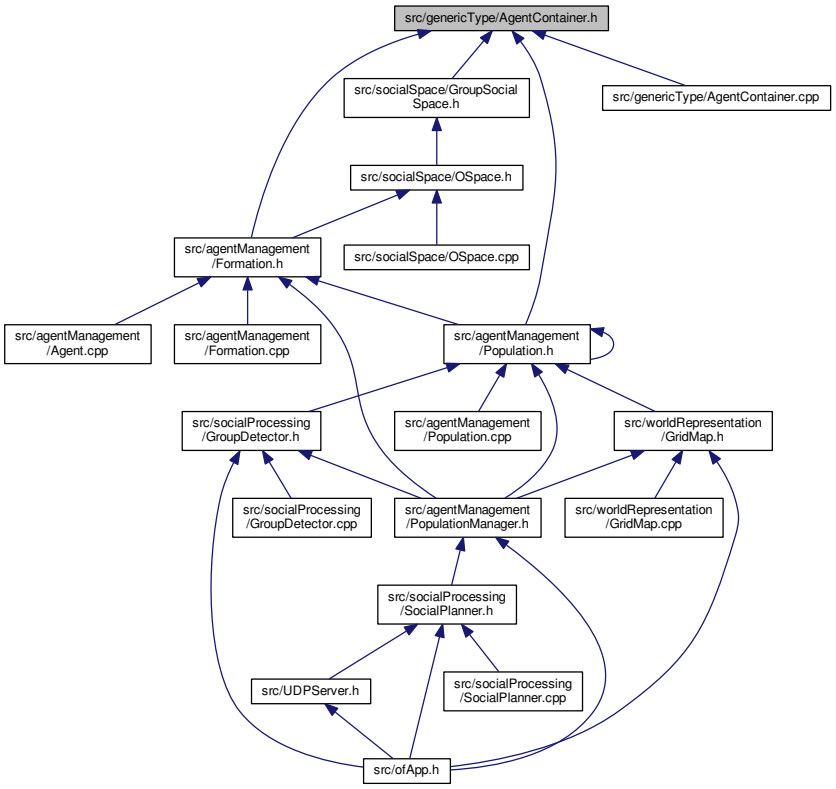
6.9 src/genericType/AgentContainer.h File Reference

```
#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

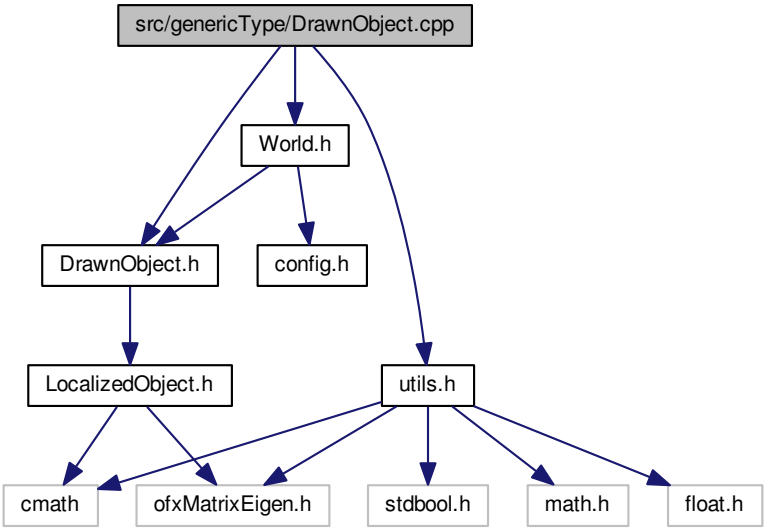
Date

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

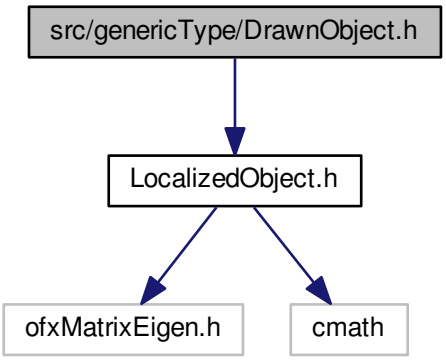
Date

27 mars 2017

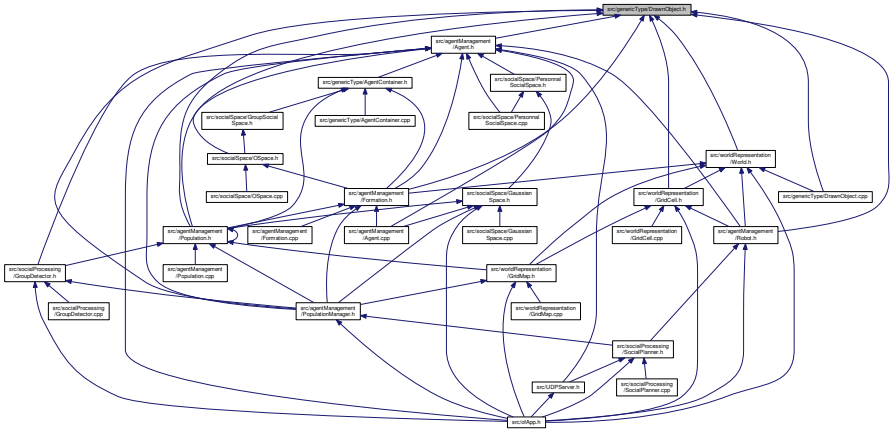
6.11 src/genericType/DrawnObject.h File Reference

```
#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

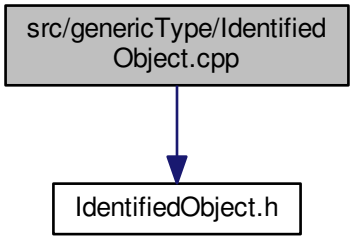
Date

27 mars 2017

6.12 src/genericType/IdentifiedObject.cpp File Reference

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

Include dependency graph for IdentifiedObject.cpp:



6.12.1 Detailed Description

Author

Paco Dupont

Version

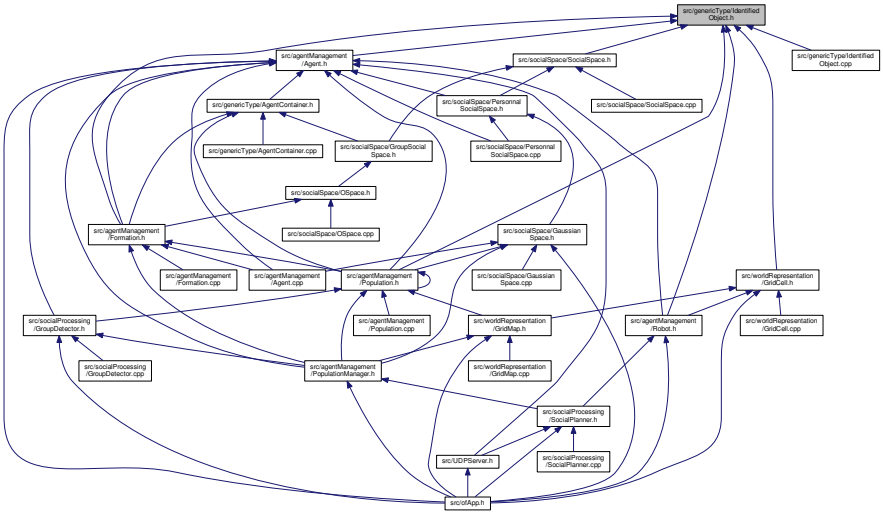
0.1

Date

27 mars 2017

6.13 src/genericType/IdentifiedObject.h File Reference

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



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

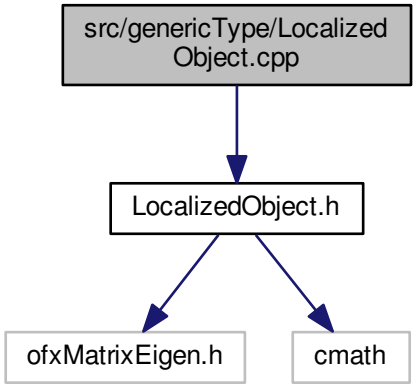
Date

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

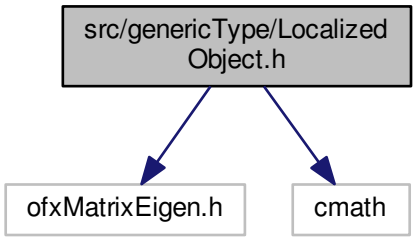
Date

27 mars 2017

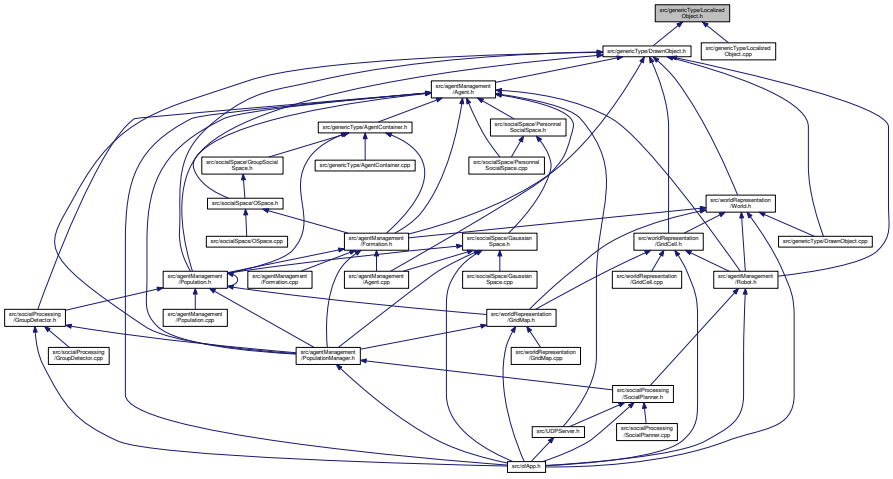
6.15 src/genericType/LocalizedObject.h File Reference

```
#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

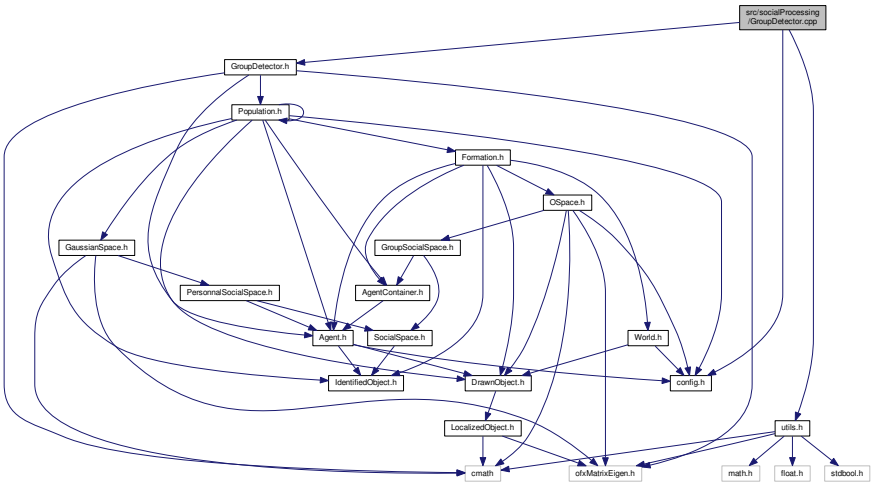
Date

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

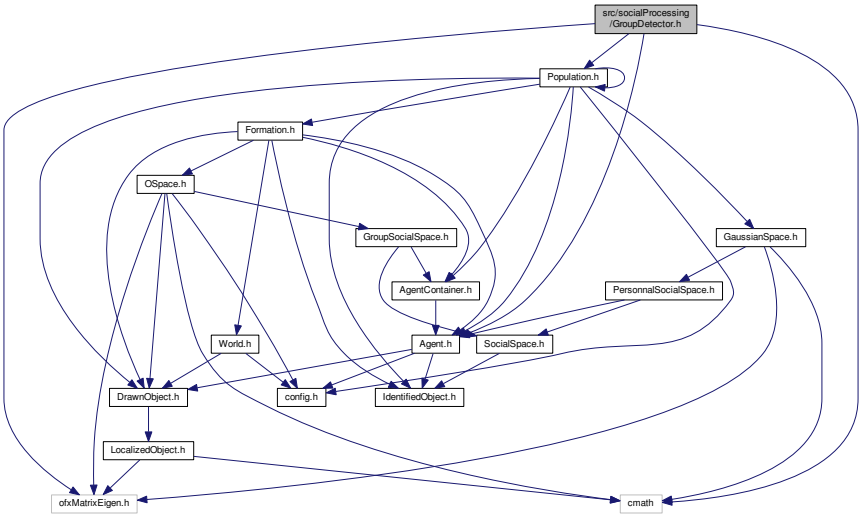
Date

10 avril 2017

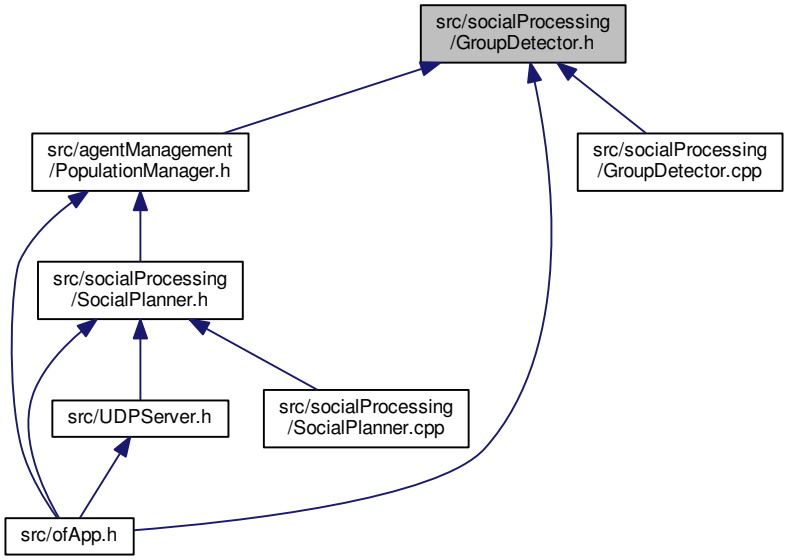
6.17 src/socialProcessing/GroupDetector.h File Reference

```
#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 class is dedicated to process every Agents in the Population and create every Formation.

6.17.1 Detailed Description

Author

Paco Dupont

Version

0.1

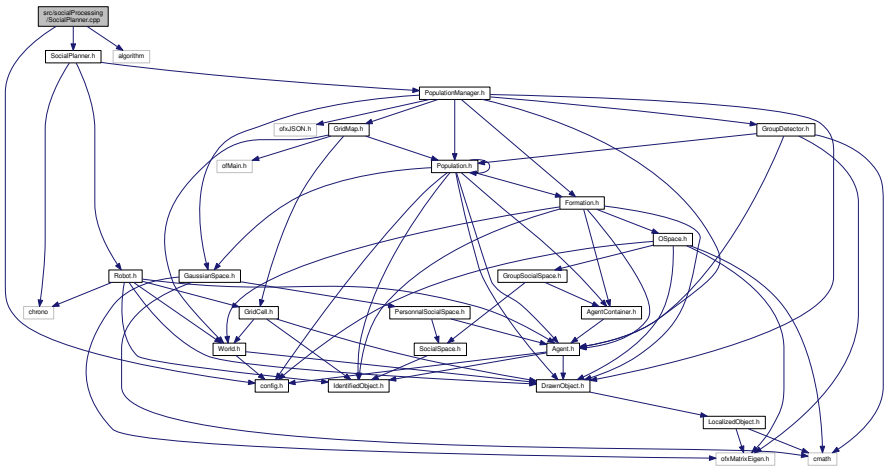
Date

10 avril 2017

6.18 src/socialProcessing/SocialPlanner.cpp File Reference

```
#include "SocialPlanner.h"
#include "config.h"
#include <algorithm>
```

Include dependency graph for SocialPlanner.cpp:



Functions

- int **min_2** (int a, int b)

6.18.1 Detailed Description

Author

Paco Dupont

Version

0.1

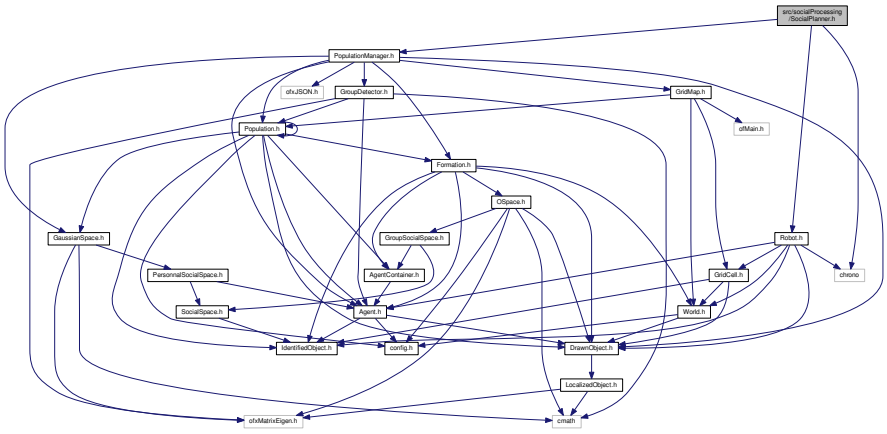
Date

7 juin 2017

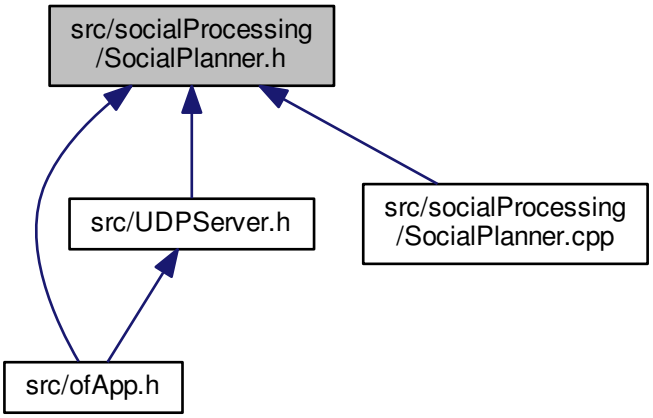
6.19 src/socialProcessing/SocialPlanner.h File Reference

```
#include "PopulationManager.h"
#include "Robot.h"
#include <chrono>
```

Include dependency graph for SocialPlanner.h:



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

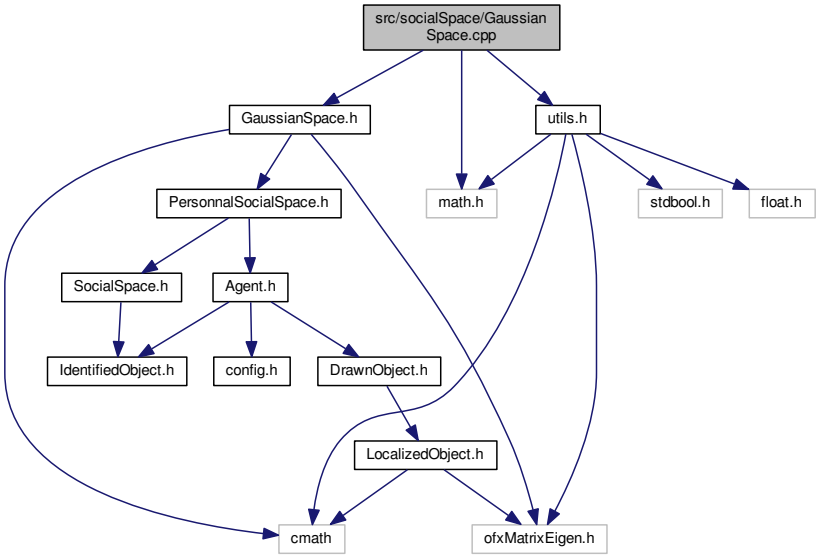
Date

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

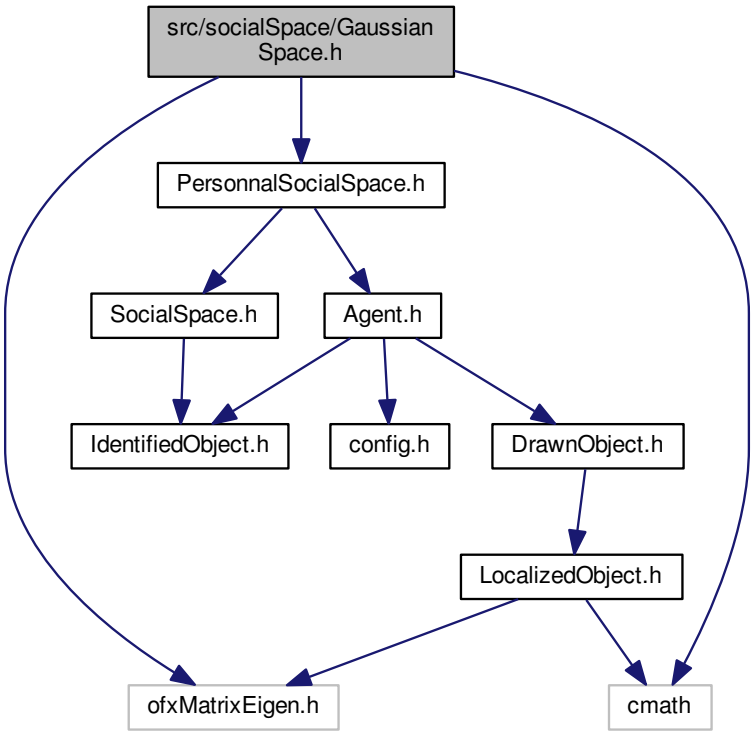
Date

28 mars 2017

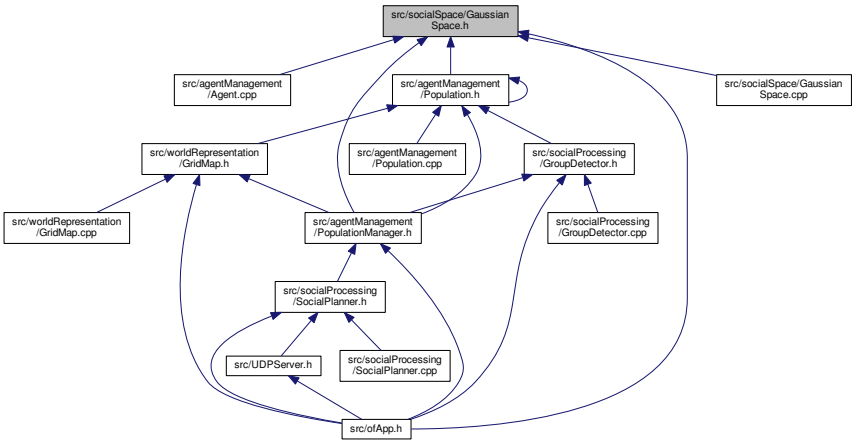
6.21 src/socialSpace/GaussianSpace.h File Reference

```
#include "PersonnalSocialSpace.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

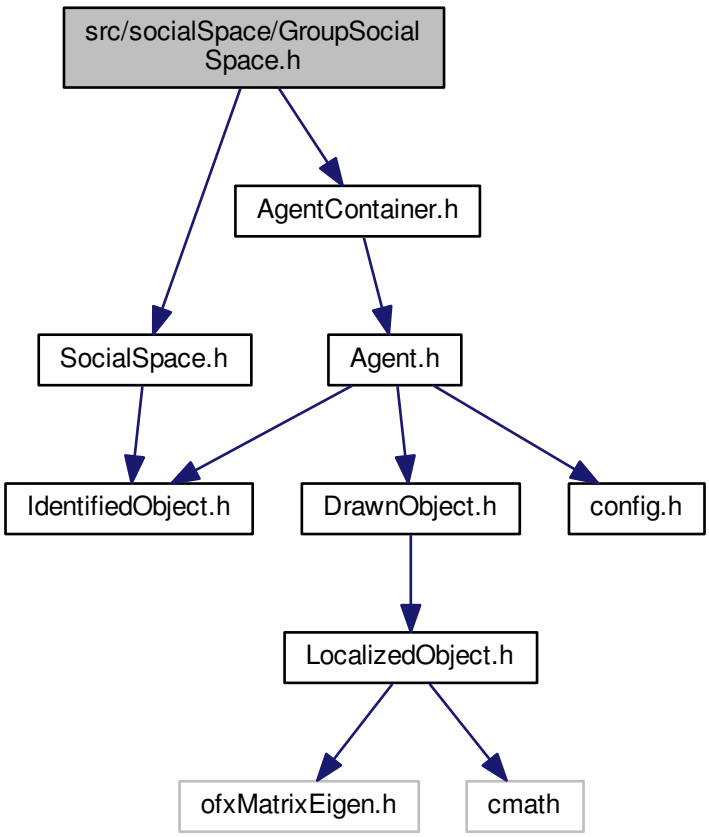
Date

28 mars 2017

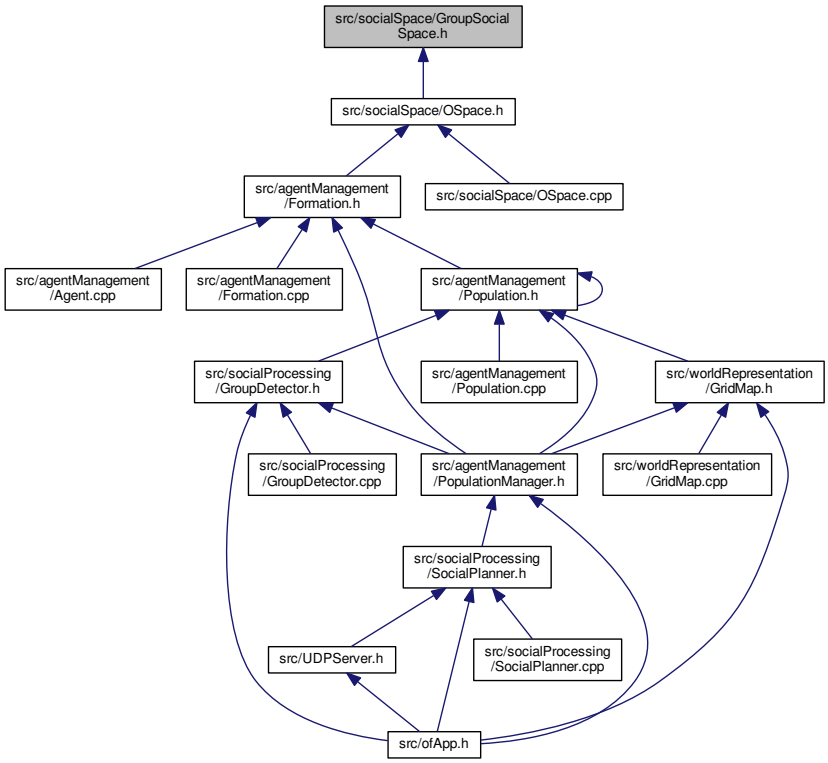
6.22 src/socialSpace/GroupSocialSpace.h File Reference

```
#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

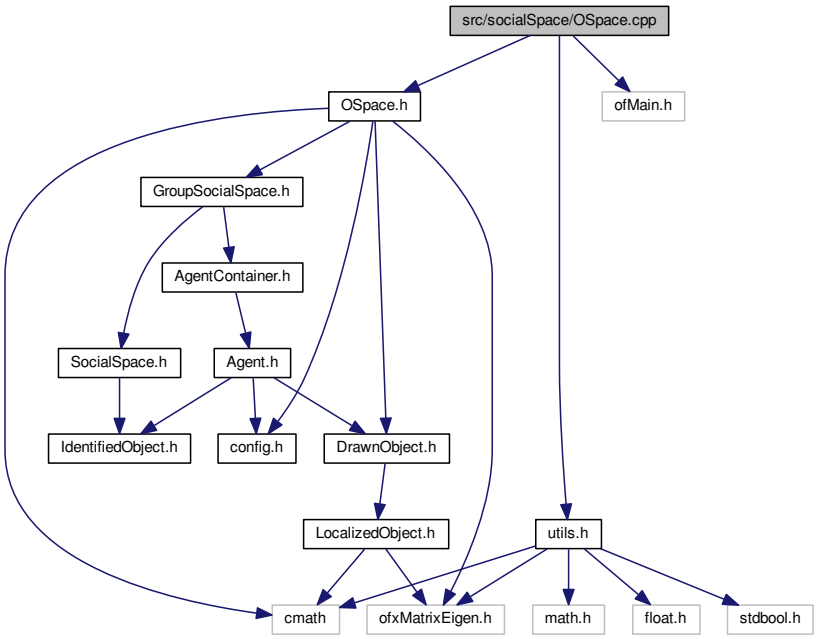
Date

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

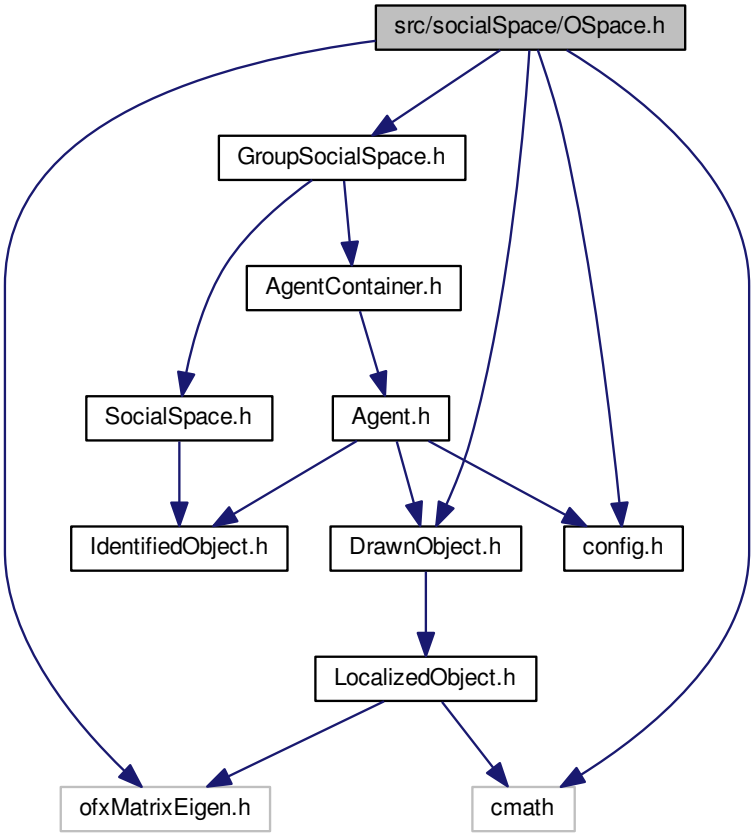
Date

27 mars 2017

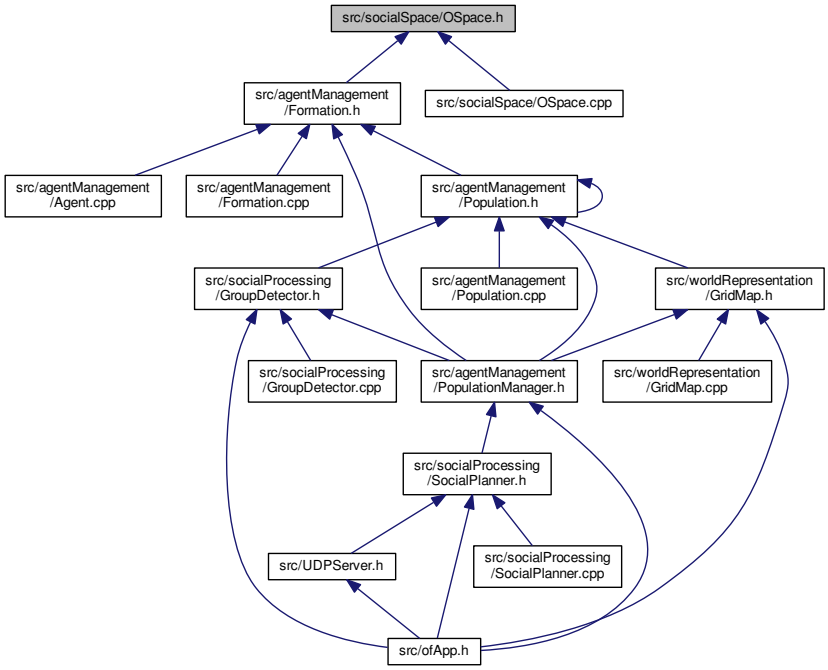
6.24 src/socialSpace/OSpace.h File Reference

```
#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

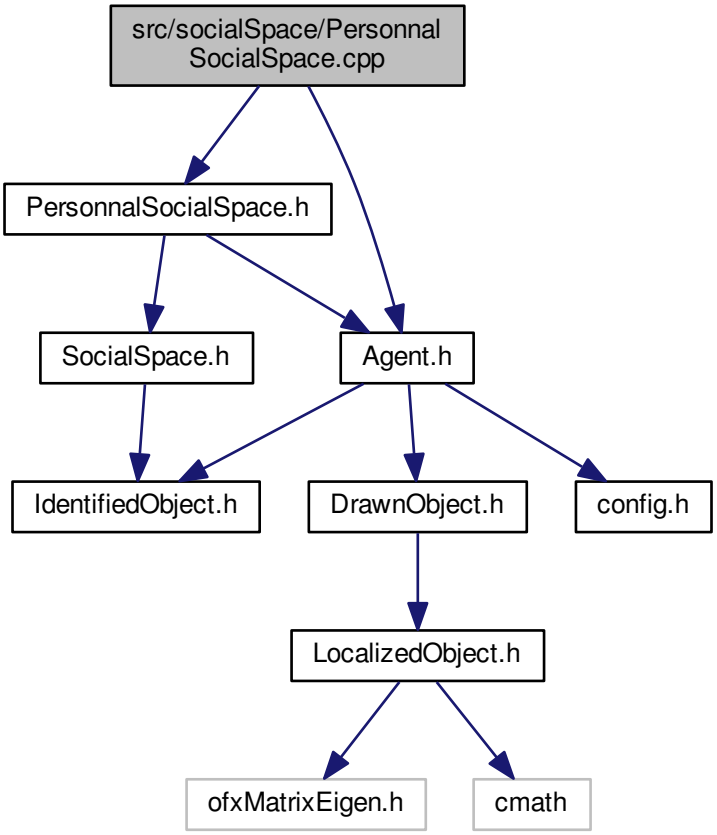
Date

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

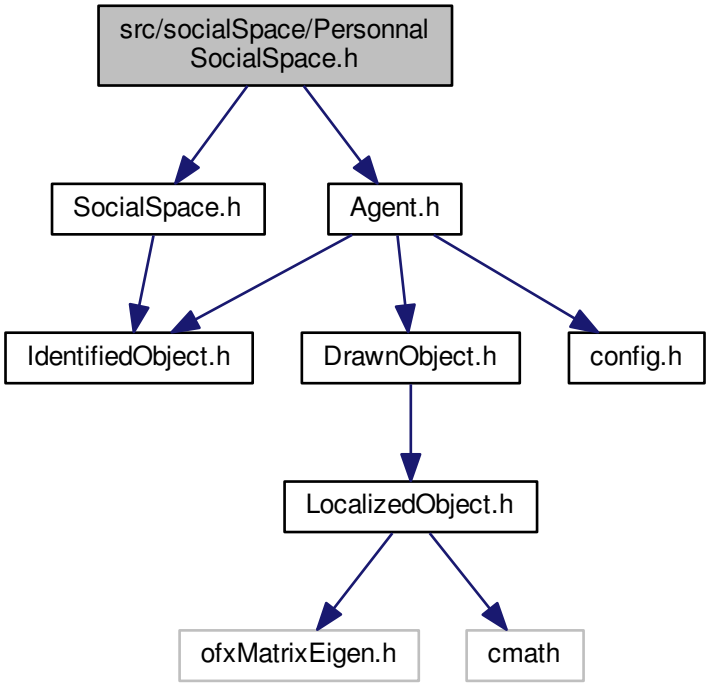
Date

29 mars 2017

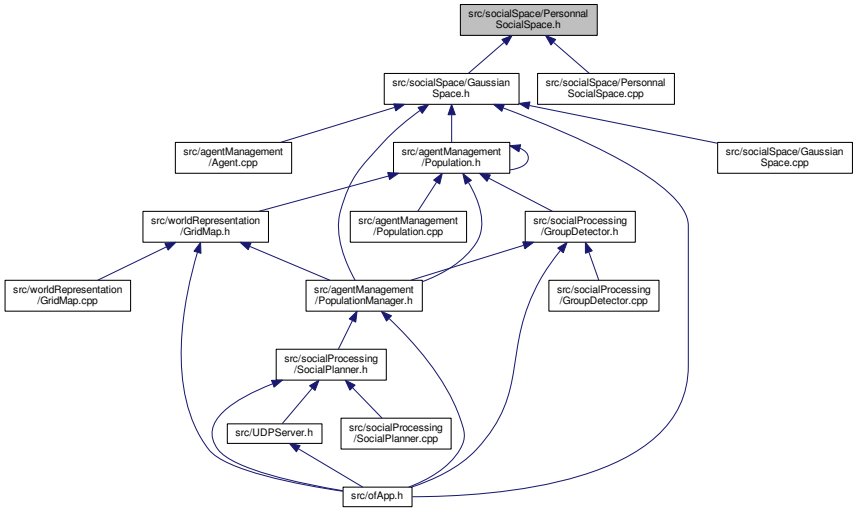
6.26 src/socialSpace/PersonnalSocialSpace.h File Reference

```
#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

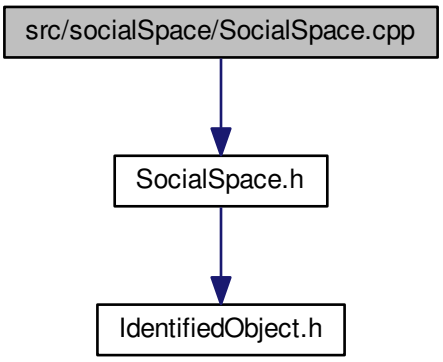
Date

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

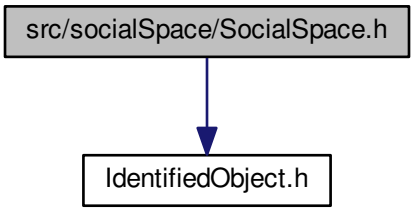
Date

27 mars 2017

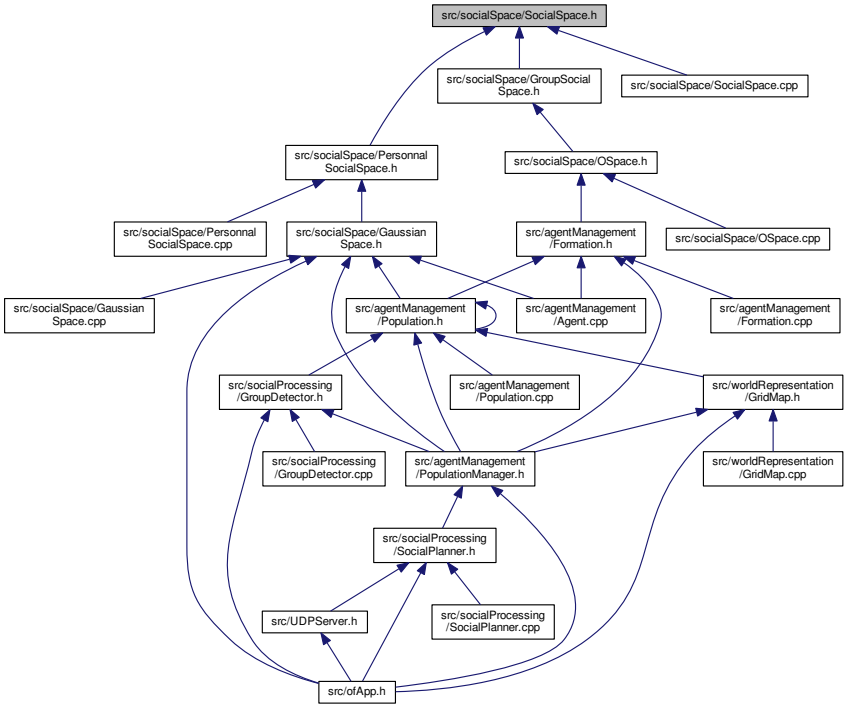
6.28 src/socialSpace/SocialSpace.h File Reference

```
#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.

6.28.1 Detailed Description

Author

Paco Dupont

Version

0.1

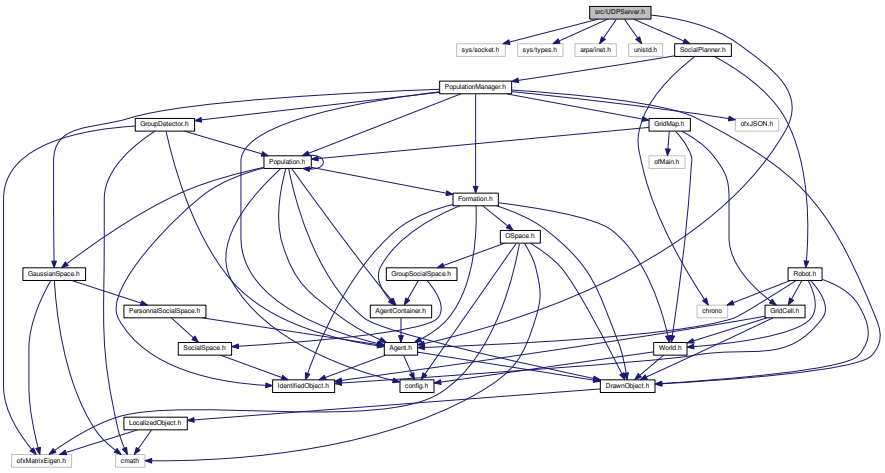
Date

27 mars 2017

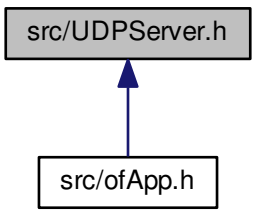
6.29 src/UDPServer.h File Reference

```
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <unistd.h>
#include "Agent.h"
#include "SocialPlanner.h"
```

Include dependency graph for UDPServer.h:



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



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

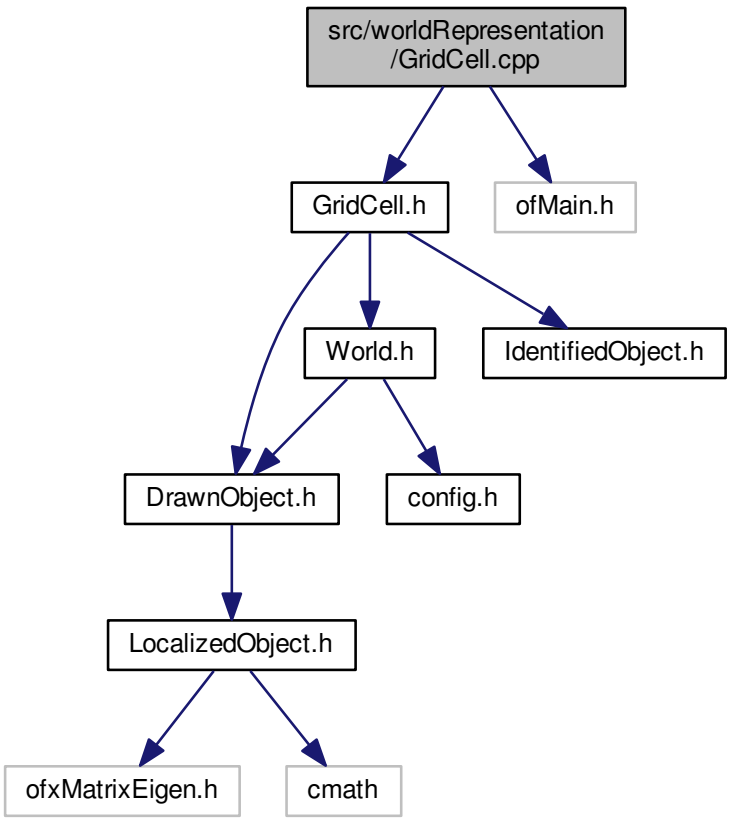
Date

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

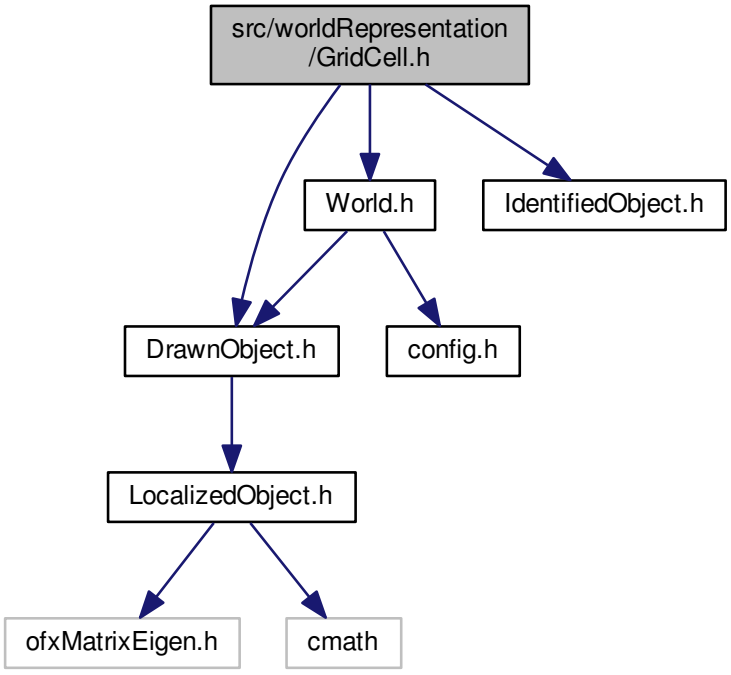
Date

6 avril 2017

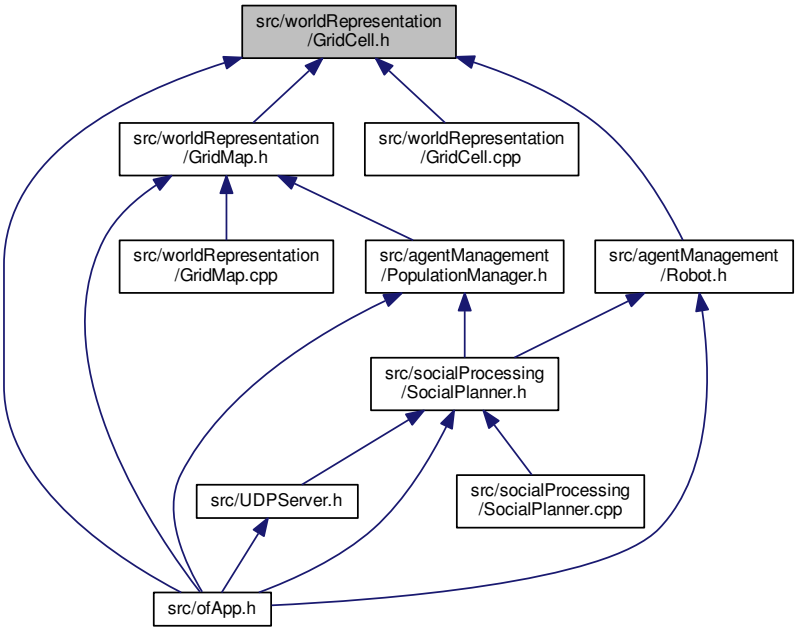
6.31 src/worldRepresentation/GridCell.h File Reference

```
#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

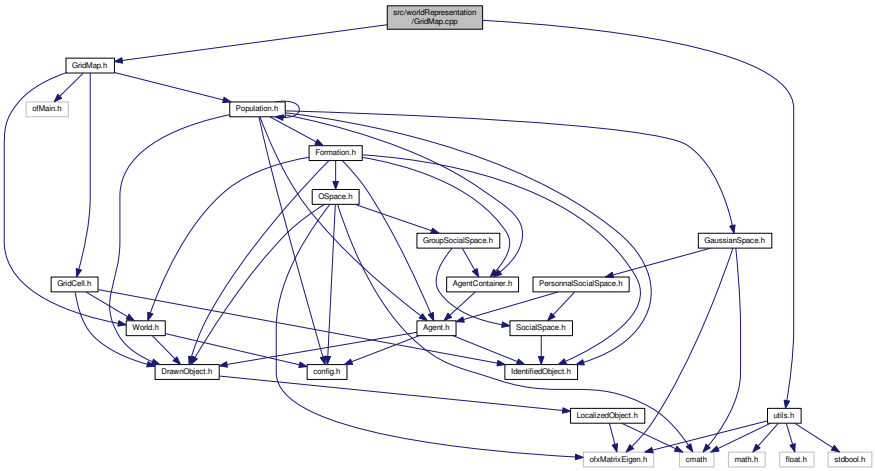
Date

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

start	: The starting GridCell
end	: 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

start	: The starting GridCell
end	: The target GridCell

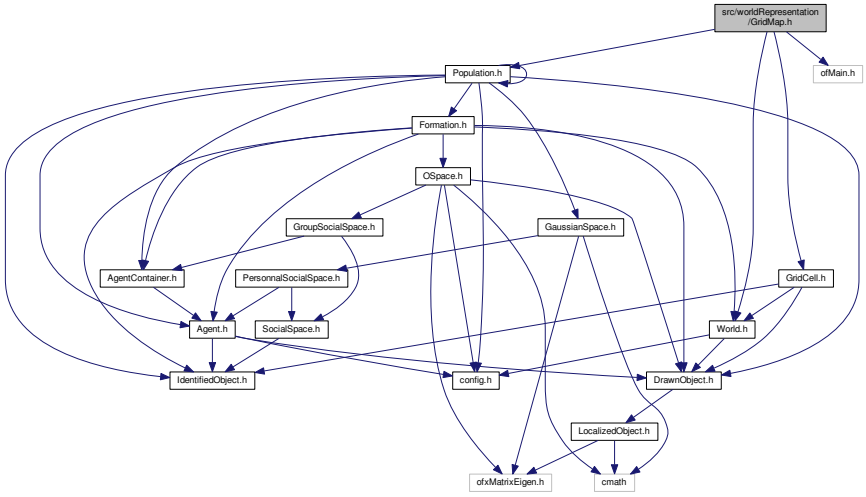
Returns

The value of the estimated cost

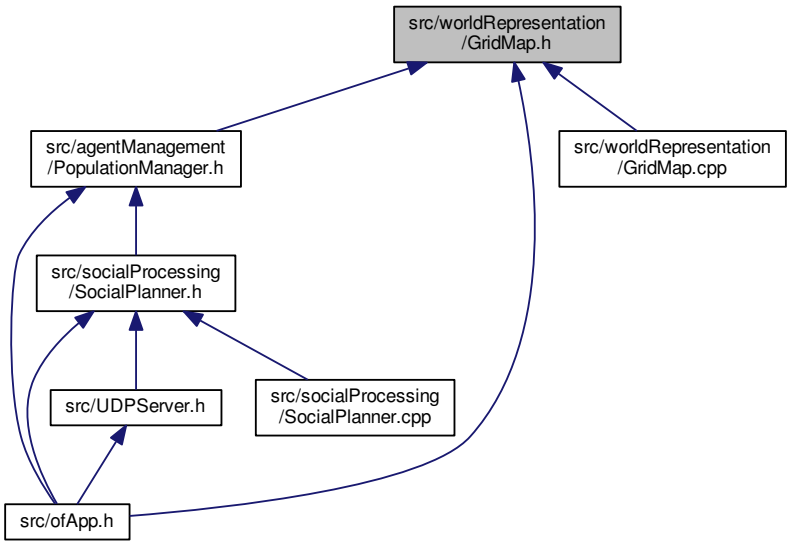
6.33 src/worldRepresentation/GridMap.h File Reference

```
#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

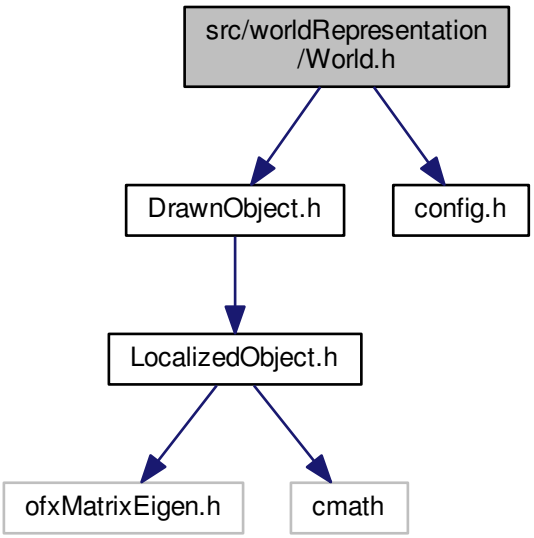
Date

6 avril 2017

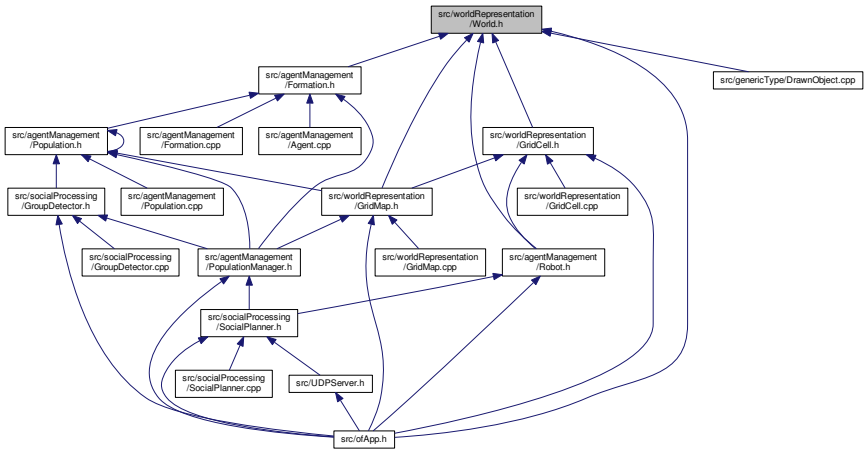
6.34 src/worldRepresentation/World.h File Reference

```
#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