ATCSimulation

1.0

Generated by Doxygen 1.7.5.1

Tue Dec 6 2011 01:57:50

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

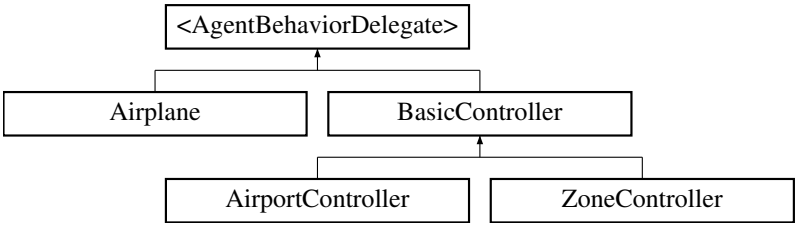
Chapter 3

Class Documentation

3.1 <AgentBehaviorDelegate> Protocol Reference

`#import <AgentBehaviorDelegate.h>`

Inheritance diagram for <AgentBehaviorDelegate>:



Public Member Functions

- (void) - analyzeMessage:withOriginalDestinator:
- (void) - startSimulation
- (void) - stopSimulation

3.1.1 Detailed Description

This protocol defines abstract methods an agent needs to implement to provide more evolved behaviors than the common BasicAgent object.

3.1.2 Member Function Documentation

3.1.2.1 - (void) analyzeMessage: dummy(NSDictionary *) *messageContent* withOriginalDestinator:(NSString *) *destinator*

Finish processing the message, as kindly cut by the BasicAgent.

Parameters

messageContent	A dictionary containing the different characteristics of the message, such as its type, its content, etc.
destinator	The original destinator of this message (the particular agent, broadcast methods, etc.).

Reimplemented in Airplane, and BasicController.

3.1.2.2 - (void) startSimulation

Asks the agent to begin to run, processing the inputs and trying to reach its goal.

Reimplemented in Airplane, and BasicController.

3.1.2.3 - (void) stopSimulation

Asks the agent to stop all the activities.

Reimplemented in Airplane, and BasicController.

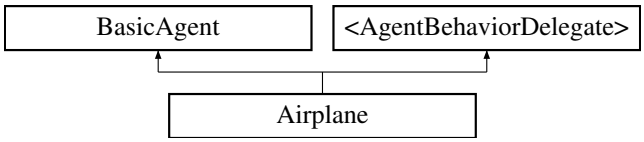
The documentation for this protocol was generated from the following file:

- ATCSimulation/Agents/AgentBehaviorDelegate.h

3.2 Airplane Class Reference

`#import <Airplane.h>`

Inheritance diagram for Airplane:



Public Member Functions

- (id) - initWithInitialData:
- (void) - **setCourse:**`[implementation]`
- (void) - **setSpeed:**`[implementation]`

- (void) - **setDestination:** [implementation]
- (void) - startSimulation [implementation]
- (void) - stopSimulation [implementation]
- (void) - analyzeMessage:withOriginalDestinator: [implementation]
- (void) - **dealloc** [implementation]

Properties

- ATCAirplaneInformation * ownInformation
- NSInteger speed
- NSInteger course
- NSString * destination
- NSString * **currentController**
- NSDate * **lastPositionCheck**

Private Member Functions

- (void) - **updatePosition** [implementation]
- (void) - **sendCurrentPosition** [implementation]
- (void) - **changeZoneWithNewController:** [implementation]

3.2.1 Detailed Description

The agent representing an airplane. It flies by itself with some characteristics defined in its ownInformation property.

3.2.2 Member Function Documentation

3.2.2.1 - (void) analyzeMessage: dummy(NSDictionary *) *messageContent* withOriginalDestinator:(NSString *) *destinator* [implementation]

Finish processing the message, as kindly cut by the BasicAgent.

Parameters

messageContent	A dictionary containing the different characteristics of the message, such as its type, its content, etc.
destinator	The original destinator of this message (the particular agent, broadcast methods, etc.).

Reimplemented from <AgentBehaviorDelegate>.

3.2.2.2 - (id) initWithInitialData: dummy(ATCAirplaneInformation *) airplaneInformation

Creates an instance of the airplane, setting the characteristics using the ATCAirplane-Information class, which holds data such as the speed, the course, etc.

Parameters

airplane-Information	A collection of informations about this airplane.

3.2.2.3 - (void) startSimulation [implementation]

Asks the agent to begin to run, processing the inputs and trying to reach its goal.

Reimplemented from <AgentBehaviorDelegate>.

3.2.2.4 - (void) stopSimulation [implementation]

Asks the agent to stop all the activities.

Reimplemented from <AgentBehaviorDelegate>.

3.2.3 Property Documentation

3.2.3.1 - (NSInteger) course [read, assign]

Gets the course of the airplane.

3.2.3.2 - (NSString *) destination [read, retain]

Gets the destination of the airplane.

3.2.3.3 - (ATCAirplaneInformation*) ownInformation [read, retain]

Gets the information instance for this object.

3.2.3.4 - (NSInteger) speed [read, assign]

Gets the speed of the airplane.

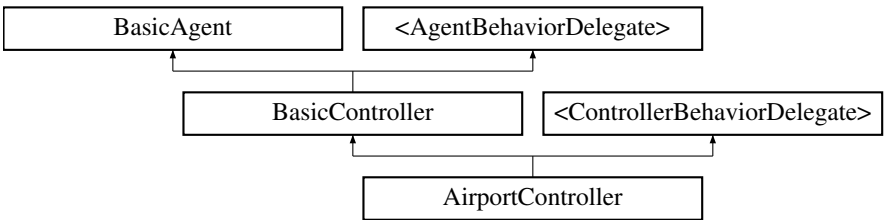
The documentation for this class was generated from the following files:

- ATCSimulation/Agents/Airplane.h
- ATCSimulation/Agents/Airplane.m

3.3 AirportController Class Reference

`#import <AirportController.h>`

Inheritance diagram for AirportController:



Public Member Functions

- (id) - initWithAirportName:andLocation:
- (void) - finishMessageAnalysis:withMessageCode:from:originallyTo: `[implementation]`

Properties

- NSString ∗ airportName
- ATCPoint ∗ airportLocation

3.3.1 Detailed Description

One of the two specialized agent playing the role of controller. The airport controller handles as the name can let it guess an airport, and the zone surrounding it. It is not able to track the planes inside the zone but can still use the default messaging ability of every agent.

3.3.2 Member Function Documentation

3.3.2.1 - (void) finishMessageAnalysis: dummy(NSString ∗) *messageContent* withMessageCode:(NVMessageCode) *code* from:(NSString ∗) *sender* originallyTo:(NSString ∗) *originalReceiver* `[implementation]`

This method is usually called if the BasicController was not able to use the message received. It transmits the message calling maybe more specific methods of the controller, with the content of the initial message.

Parameters

message-Content	The content of the message, containing formatted information.
code	The code of the message.
sender	The initial sender of the message.
original-Receiver	The destination of the message, which can this agent, or all the agents etc.

Reimplemented from <ControllerBehaviorDelegate>.

3.3.2.2 - (id) initWithAirportName: dummy(NSString *) *airportName* andLocation:(ATCPoint *) *airportLocation*

Creates one instance of the agent, with some specific attributes.

Parameters

airportName	The name of the airport, used by the airplanes to track their destination.
airport-Location	The position of the runway.

3.3.3 Property Documentation

3.3.3.1 - (ATCPoint*) airportLocation [read, retain]

Gets the position of the runway.

3.3.3.2 - (NSString*) airportName [read, retain]

Gets the name of the airport.

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

- ATCSimulation/Agents/AirportController.h
- ATCSimulation/Agents/AirportController.m

3.4 AppDelegate Class Reference

Public Member Functions

- (void) - **dealloc** [implementation]
- (BOOL) - **application:didFinishLaunchingWithOptions:** [implementation]

- (void) - **applicationWillResignActive:** [implementation]
- (void) - **applicationDidEnterBackground:** [implementation]

- (void) - **applicationWillEnterForeground:**[implementation]
- (void) - **applicationDidBecomeActive:**[implementation]
- (void) - **applicationWillTerminate:**[implementation]

Properties

- IBOutlet UIWindow ∗ **window**
- IBOutlet MainInterfaceController ∗ **viewController**

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

- ATCSimulation/AppDelegate.h
- ATCSimulation/AppDelegate.m

3.5 Artifacts Class Reference

```
#import <Artifacts.h>
```

Static Public Member Functions

- (NSInteger) + calculateCurrentZonefromPoint:
- (float) + distanceFromNextZone:onRoute:
- (ATCPoint ∗) + calculateNewPositionFromCurrent:afterInterval:

3.5.1 Detailed Description

Some other methods, that don't need any specific link to an actual instance of the objects, explaining why they all are static.

3.5.2 Member Function Documentation

3.5.2.1 + (NSInteger) calculateCurrentZonefromPoint: dummy(ATCPoint ∗) *location*

Returns the id of the current zone where the plane is located.

Parameters

location	The position of the airplane.

Returns

The ID of the zone where the plane is at.

3.5.2.2 + (ATCPoint ∗) calculateNewPositionFromCurrent: dummy(ATCAirplaneInformation ∗) *currentPosition* afterInterval:(NSTimeInterval) *interval*

Convenient method to calculate the new point reached by the airplane after flying for a certain time, with the parameters of the flight given in the other parameter.

Parameters

currentPosition	The information about the current airplane, such as the route, the speed, and the initial position.
interval	The length of the flight.

Returns

Returns the new location of the airplane.

3.5.2.3 + (float) distanceFromNextZone: dummy(ATCPoint ∗) *position* onRoute:(NSInteger ∗) *route*

Calculates the distance to the next zone with the route and the initial position.

Parameters

position	The point where the airplane currently is located.
route	The azimut it follows.

Returns

The distance to the next zone on a straight line.

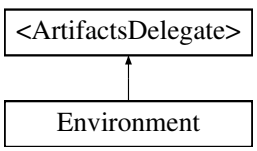
The documentation for this class was generated from the following files:

- ATCSimulation/Agents/Utils/Artifacts.h
- ATCSimulation/Agents/Utils/Artifacts.m

3.6 <ArtifactsDelegate> Protocol Reference

```
#import <ArtifactsDelegate.h>
```

Inheritance diagram for <ArtifactsDelegate>:



Public Member Functions

- (void) - updateInterfaceWithInformationsForZone:
- (void) - landAirplane:
- (void) - crashAirplane:

3.6.1 Detailed Description

Protocol declaring abstract methods that can be used by the agents. These methods, or artifacts, are implemented by the environment, and provide some kind of services at disposition.

3.6.2 Member Function Documentation

3.6.2.1 - (void) crashAirplane: dummy(ATCAirplaneInformation *) airplane

Asks the environment to hide the specified airplane, as it crashed (after running out of fuel or colliding with another airplane).

Parameters

airplane	The informations about the airplane that just had an accident.

Reimplemented in Environment.

3.6.2.2 - (void) landAirplane: dummy(ATCAirplaneInformation *) airplane

Asks the environment to hide the specified airplane, as it has reached its destination.

Parameters

airplane	The informations about the airplane that should land.

Reimplemented in Environment.

3.6.2.3 - (void) updateInterfaceWithInformationsForZone: dummy(NSArray *) informations

Asks the environment to display on the map the informations retrieved by the controllers.

Parameters

informations	An array containing ATCAirplaneInformation objects, as created by the caller.

Reimplemented in Environment.

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

- ATCSimulation/Agents/Utils/ArtifactsDelegate.h

3.7 ATCAirplaneInformation Class Reference

`#import <ATCAirplaneInformation.h>`

Public Member Functions

- (id) - **initWithZone:andPoint:**
- (void) - **dealloc** `[implementation]`

Static Public Member Functions

- (ATCAirplaneInformation ∗) + planeInformationFromExisting:

Properties

- NSInteger currentZoneID
- ATCPoint ∗ coordinates
- NSInteger speed
- NSInteger course
- NSString ∗ destination
- NSString ∗ airplaneName

3.7.1 Detailed Description

A class containing all the information needed to completely characterize an airplane. It is used both by the airplane and the controllers to hold the information about the airplane, and to be able to represent it correctly then.

3.7.2 Member Function Documentation

3.7.2.1 + (ATCAirplaneInformation ∗) planeInformationFromExisting: dummy(ATCAirplaneInformation ∗) *info*

Duplicates one object.

3.7.3 Property Documentation

3.7.3.1 - (NSString∗) airplaneName `[read, write, retain]`

The name of the airplane, which is unique.

3.7.3.2 - (ATCPoint*) coordinates [read, write, retain]

Property describing the location of the airplane.

3.7.3.3 - (NSInteger) course [read, write, assign]

Its route.

3.7.3.4 - (NSInteger) currentZoneID [read, write, assign]

Property containing the zone the airplane is in.

3.7.3.5 - (NSString*) destination [read, write, retain]

Its destination.

3.7.3.6 - (NSInteger) speed [read, write, assign]

The speed of the airplane.

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

- ATCSimulation/Agents/Utils/ATCAirplaneInformation.h
- ATCSimulation/Agents/Utils/ATCAirplaneInformation.m

3.8 ATCPoint Class Reference

```
#import <ATCPoint.h>
```

Public Member Functions

- (id) - initWithCoordinateX:andCoordinateY:

Static Public Member Functions

- (ATCPoint *) + pointFromExisting:

Properties

- float X
- float Y

3.8.1 Detailed Description

A class containing a point on the map.

3.8.2 Member Function Documentation

3.8.2.1 - (id) initWithCoordinateX: dummy(float) *x* andCoordinateY:(float) *y*

Creates a point at the specified coordinates.

Parameters

x	
y	

3.8.2.2 + (**ATCPoint** ∗) pointFromExisting: dummy(**ATCPoint** ∗) *point*

Factory method to duplicate a point.

Parameters

The	point to duplicate.

Returns

A point having the same characteristics as the initial one.

3.8.3 Property Documentation

3.8.3.1 - (float) X [read, write, assign]

The x coordinate of the point.

3.8.3.2 - (float) Y [read, write, assign]

The y coordinate of the point.

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

- ATCSimulation/Agents/Utils/ATCPoint.h
- ATCSimulation/Agents/Utils/ATCPoint.m

3.9 ATCZone Class Reference

```
#import <ATCZone.h>
```

Public Member Functions

- (id) - initWithCorners:withControllerName:andIsAirport:
- (void) - addAdjacentZone:
- (float) - calculateDistanceToZoneBorderWithPosition:
- (BOOL) - pointBelongsToZone:
- (void) - **dealloc** `[implementation]`

Properties

- BOOL airport
- NSString ∗ controllerName
- NSMutableSet ∗ adjacentZones
- NSArray ∗ corners
- NSMutableArray ∗ borders

3.9.1 Detailed Description

A class representing a zone in the environment. A zone if composed of borders, which are segments instance of the class ZoneBorderSegment. Each zone has a controller, which is either an AirportController or a ZoneController.

3.9.2 Member Function Documentation

3.9.2.1 - (void) addAdjacentZone: dummy(ATCZone ∗) *zone*

Method to add a neighbour to the zone.

Parameters

zone	The zone which shares a border with the current one.

3.9.2.2 - (float) calculateDistanceToZoneBorderWithPosition: dummy(ATCAirplaneInformation ∗) *position*

A method to know the distance from the airplane to the nearest frontier of the zone.

Parameters

position	The information about the airplane, with useful data such as the current position and the course of the airplane.

Returns

Returns the shortest (straight) distance to the zone, depending on the course of the airplane.

3.9.2.3 - (id) initWithCorners: dummy(NSArray ∗) *cornersArray* withControllerName:(NSString ∗) *controllerName* andIsAirport:(BOOL) *airport*

Creates an instance of the zone, with the specified corners, and the name of the controller hosted inside.

Parameters

cornersArray	An array of all the extremities forming the polygone. These corners are then analyzed by the instance to create the corresponding frontiers.
controller-Name	The name of the controller hosted inside the zone.
airport	A boolean telling if the current zone has an airway or not, setting the type of controller which is inside the zone.

3.9.2.4 - (BOOL) pointBelongsToZone: dummy(ATCPoint ∗) *point*

Tests if a point is inside the zone.

Parameters

point	The point to test.

Returns

Returns YES if the point is inside the zone.

3.9.3 Property Documentation

3.9.3.1 - (NSMutableSet∗) adjacentZones [read, retain]

Gets the zones next to this one.

3.9.3.2 - (BOOL) airport [read, assign]

Gets the type of controller inside the zone.

3.9.3.3 - (NSMutableArray∗) borders [read, retain]

Gets the borders of the zone.

3.9.3.4 - (NSString∗) controllerName [read, retain]

Gets the name of the controller.

3.9.3.5 - (NSArray∗) corners [read, retain]

Gets the corners of the zone.

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

- ATCSimulation/Agents/Utils/ATCZone.h
- ATCSimulation/Agents/Utils/ATCZone.m

3.10 ATCZoneBorderSegment Class Reference

`#import <ATCZoneBorderSegment.h>`

Public Member Functions

- (id) - initWithExtremity1:andExtremity2:withDirectionPositive:
- (BOOL) - pointBelongsToGeneratedHalfSpace:
- (float) - calculateDistanceToSegment:
- (void) - **dealloc** [implementation]

Properties

- ATCPoint ∗ **extremity1**
- ATCPoint ∗ **extremity2**
- float **aLine**
- float **bLine**
- float **cLine**
- BOOL **directionPositive**
- float **aOrthogonalLine1**
- float **bOrthogonalLine1**
- float **cOrthogonalLine1**
- float **aOrthogonalLine2**
- float **bOrthogonalLine2**
- float **cOrthogonalLine2**

Private Member Functions

- (BOOL) - **testHalfSpaceWithInequationCoefficientsA:andB:andC:and-InequalityPositive:atPoint:** [implementation]

3.10.1 Detailed Description

A class to represent one element of the border, containing methods to perform atomic operations on this segment. It contains properties to obtain the representation of the line, with the equation ay + bx + c = 0.

3.10.2 Member Function Documentation

3.10.2.1 - (float) calculateDistanceToSegment: dummy(ATCAirplaneInformation ∗) *testedPosition*

Calculates the distance from the location of the airplane to the current segment, distance which can be infinite if the airplane never meets the segment on its current course.

Parameters

testedPosition	The information about the aircraft to test.

Returns

Returns the distance to the current segment, if the airplane continued to fly straight on its course.

3.10.2.2 - (id) initWithExtremity1: dummy(ATCPoint ∗) *extremity1* andExtremity2:(ATCPoint ∗) *extremity2* withDirectionPositive:(BOOL) *positive*

Creates one segment, from extremity1 to extremity2.

Parameters

extremity1	The first extremity of the segment.
extremity2	The second extremity of the segment.
positive	A boolean telling if the zone is above or under this segment, useful to determine the direction of the half-space.

3.10.2.3 - (BOOL) pointBelongsToGeneratedHalfSpace: dummy(ATCPoint ∗) *testedPoint*

Tests if the point belongs to the half-space generated by the inequation, with the coefficients expressed in the other properties.

Parameters

testedPoint	The point to test.

Returns

Returns YES if the point is inside the half-space.

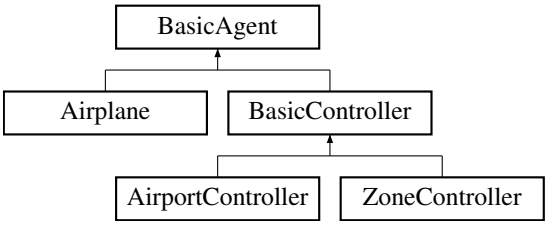
The documentation for this class was generated from the following files:

- ATCSimulation/Agents/Utils/ATCZoneBorderSegment.h
- ATCSimulation/Agents/Utils/ATCZoneBorderSegment.m

3.11 BasicAgent Class Reference

`#import <BasicAgent.h>`

Inheritance diagram for BasicAgent:



Public Member Functions

- (id) - initWithAgentName:
- (void) - sendMessage:fromType:toAgent:
- (void) - receiveMessage:
- (void) - **dealloc** `[implementation]`

Properties

- id< AgentBehaviorDelegate > agentBehaviorDelegate
- NSString * agentName

3.11.1 Detailed Description

A basic class defining some ground behaviors for an agent, such as the messaging capabilities. As an agent it replies to some of the messages received from the other agents, using its reasoning capabilities to determine if an answer is needed and wanted according to its goal.

3.11.2 Member Function Documentation

3.11.2.1 - (id) initWithAgentName: dummy(NSString *) *name*

Creates an instance of an agent.

Parameters

name	The name of the agent, setting the destinator for the messages.

3.11.2.2 - (void) receiveMessage: dummy(NSNotification ∗) *notification*

The method which is called when the agent receives a message from others.

Parameters

notification	The object wrapping the different elements of the message.

3.11.2.3 - (void) sendMessage: dummy(NSString ∗) *message* fromType:(NVMessageCode) *type* toAgent:(NSString ∗) *agentName*

Top level method to send messages to other agents.

Parameters

message	The content of the message to be sent, as a string.
type	The corresponding code of the message.
agentName	The name of the destinator of this message.

3.11.3 Property Documentation

3.11.3.1 - (id<AgentBehaviorDelegate>) agentBehaviorDelegate [read, write, retain]

A property used to set the object implementing the various methods of the AgentBehaviorDelegate.

3.11.3.2 - (NSString∗) agentName [read, retain]

Gets the name of this agent.

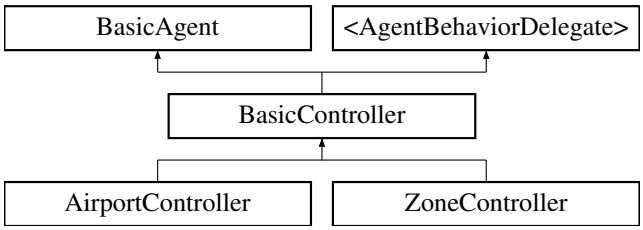
The documentation for this class was generated from the following files:

- ATCSimulation/Agents/BasicAgent.h
- ATCSimulation/Agents/BasicAgent.m

3.12 BasicController Class Reference

`#import <BasicController.h>`

Inheritance diagram for BasicController:



Public Member Functions

- (void) - detectAirplanesInZone
- (id) - **init** `[implementation]`
- (void) - **dealloc** `[implementation]`
- (void) - startSimulation `[implementation]`
- (void) - stopSimulation `[implementation]`
- (void) - analyzeMessage:withOriginalDestinator: `[implementation]`

Static Public Member Functions

- (int) + createZoneID
- (NSString ∗) + zoneIdentifierAsStringWithID:
- (NSString ∗) + messageIdentifierForZone:

Properties

- id< ControllerBehaviorDelegate > controllerDelegate
- NSMutableDictionary ∗ controlledAirplanes
- int zoneID

3.12.1 Detailed Description

A class defining the common behaviors of the specialized controllers. It also uses another delegate to finish handling messages, start and stop the simulation, etc.

3.12.2 Member Function Documentation

3.12.2.1 - (void) analyzeMessage: dummy(NSDictionary ∗) *messageContent* withOriginalDestinator:(NSString ∗) *destinator* `[implementation]`

Finish processing the message, as kindly cut by the BasicAgent.

Parameters

messageContent	A dictionary containing the different characteristics of the message, such as its type, its content, etc.
destinator	The original destinator of this message (the particular agent, broadcast methods, etc.).

Reimplemented from <AgentBehaviorDelegate>.

3.12.2.2 + (int) createZoneID

An abstract method to create a unique ID for each class.

Returns

Returns an unique id that can be used to identify the zones.

3.12.2.3 - (void) detectAirplanesInZone

The active radar mode, trying to recover information about the airplanes currently flying in the zone.

3.12.2.4 + (NSString *) messageIdentifierForZone: dummy(int) *zoneID*

Another convenient method to create the the identifier the zone is listening to for incoming messages.

3.12.2.5 - (void) startSimulation [implementation]

Asks the agent to begin to run, processing the inputs and trying to reach its goal.

Reimplemented from <AgentBehaviorDelegate>.

3.12.2.6 - (void) stopSimulation [implementation]

Asks the agent to stop all the activities.

Reimplemented from <AgentBehaviorDelegate>.

3.12.2.7 + (NSString *) zoneIdentifierAsStringWithID: dummy(int) *ID*

Convenient method to represent the zone ID, used as agent name.

Parameters

id	The id of the zone where a representation is needed.

3.12.3 Property Documentation

3.12.3.1 - (NSMutableDictionary*) controlledAirplanes [read, retain]

A dictionary referencing the different airplanes controlled by this controller, containing the name of the agent as key and the information about the airplane as value.

3.12.3.2 - (id<ControllerBehaviorDelegate>) controllerDelegate [read, write, retain]

The delegate implementing the various specialized behaviors of a controller.

3.12.3.3 - (int) zoneID [read, assign]

The zone the controller is belonging to.

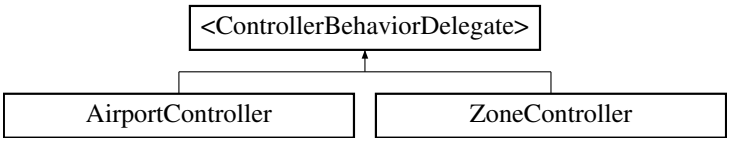
The documentation for this class was generated from the following files:

- ATCSimulation/Agents/BasicController.h
- ATCSimulation/Agents/BasicController.m

3.13 <ControllerBehaviorDelegate> Protocol Reference

`#import <ControllerBehaviorDelegate.h>`

Inheritance diagram for <ControllerBehaviorDelegate>:



Public Member Functions

- (void) - finishMessageAnalysis:withMessageCode:from:originallyTo:

3.13.1 Detailed Description

This protocol defines the various methods a controller needs to implement, to provide an extended support for other capabilities reached through the messaging features.

3.13.2 Member Function Documentation

3.13.2.1 - (void) finishMessageAnalysis: dummy(NSString ∗) *messageContent* withMessageCode:(NVMessageCode) *code* from:(NSString ∗) *sender* originallyTo:(NSString ∗) *originalReceiver*

This method is usually called if the BasicController was not able to use the message received. It transmits the message calling maybe more specific methods of the controller, with the content of the initial message.

Parameters

messageContent	The content of the message, containing formatted information.
code	The code of the message.
sender	The initial sender of the message.
originalReceiver	The destination of the message, which can this agent, or all the agents etc.

Reimplemented in AirportController, and ZoneController.

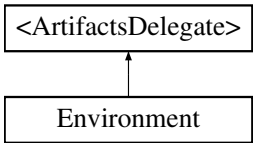
The documentation for this protocol was generated from the following file:

- ATCSimulation/Agents/ControllerBehaviorDelegate.h

3.14 Environment Class Reference

`#import <Environment.h>`

Inheritance diagram for Environment:



Public Member Functions

- (id) - initWithDisplayDelegate:
- (void) - startSimulation
- (void) - stopSimulation
- (void) - resetSimulation
- (void) - updateInterfaceWithInformationsForZone:`[implementation]`
- (void) - landAirplane:`[implementation]`
- (void) - crashAirplane:`[implementation]`

Properties

- id< EnvironmentDisplayDelegate > displayDelegate
- NSMutableArray * zones
- NSMutableArray * airportControllers
- NSMutableArray * zoneControllers
- NSMutableArray * airplanes
- NSTimer * **displayUpdateTimer**

Private Member Functions

- (void) - **createEnvironment** `[implementation]`
- (Airplane *) - **createAirplaneWithInitialInfo:** `[implementation]`
- (void) - **askForDisplayUpdate:** `[implementation]`
- (void) - **performDisplayUpdate** `[implementation]`
- (void) - **performAddAirplaneToMap:** `[implementation]`
- (void) - **performAddMultipleAirplanesToMap** `[implementation]`
- (void) - **performAirplane:** `[implementation]`
- (void) - **representStartingEnvironment** `[implementation]`

3.14.1 Detailed Description

The environment of the simulation, handling the different agents that interact together, the playground for them (the map, the borders, the zones).

It provides several artifacts to these agents, defined either in the Artifacts or in the -ArtifactsDelegate, to help them perform certain actions (such as calculate their position, etc.), or access the interface.

The environment also owns a reference to the main interface of the application, so that it can display on the screen information to the user.

3.14.2 Member Function Documentation

3.14.2.1 - (void) crashAirplane: dummy(ATCAirplaneInformation *) *airplane* `[implementation]`

Asks the environment to hide the specified airplane, as it crashed (after running out of fuel or colliding with another airplane).

Parameters

airplane	The informations about the airplane that just had an accident.

Reimplemented from <ArtifactsDelegate>.

3.14.2.2 - (id) initWithDisplayDelegate: dummy(id) *object*

Creates an instance of the environment, and sets the display delegate responding to the methods defined in the EnvironmentDisplayDelegate protocol, so that the environment can ask to perform certain actions on the interface.

Parameters

The	delegate allowing access to the interface. All calls to the interface must be made on the main thread.

3.14.2.3 - (void) landAirplane: dummy(**ATCAirplaneInformation** ∗) *airplane* [implementation]

Asks the environment to hide the specified airplane, as it has reached its destination.

Parameters

airplane	The informations about the airplane that should land.

Reimplemented from <ArtifactsDelegate>.

3.14.2.4 - (void) resetSimulation

Resets the simulation, recreates the environment and the agents interacting in it.

3.14.2.5 - (void) startSimulation

Starts the simulation once it is ready.

3.14.2.6 - (void) stopSimulation

Stops the simulation.

3.14.2.7 - (void) updateInterfaceWithInformationsForZone: dummy(NSArray ∗) *informations* [implementation]

Asks the environment to display on the map the informations retrieved by the controllers.

Parameters

informations	An array containing ATCAirplaneInformation objects, as created by the caller.

Reimplemented from <ArtifactsDelegate>.

3.14.3 Property Documentation

3.14.3.1 - (NSMutableArray*) airplanes [read, retain]

Gets the airplanes running in the simulation.

3.14.3.2 - (NSMutableArray*) airportControllers [read, retain]

Gets the airport controllers running in the simulation.

3.14.3.3 - (id<EnvironmentDisplayDelegate>) displayDelegate [read, write, retain]

Property permitting an access to the interface delegate.

3.14.3.4 - (NSMutableArray*) zoneControllers [read, retain]

Gets the zone controllers running in the simulation.

3.14.3.5 - (NSMutableArray*) zones [read, retain]

Gets the zones that cluster the map.

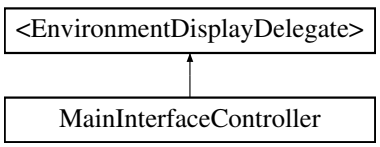
The documentation for this class was generated from the following files:

- ATCSimulation/Agents/Environment.h
- ATCSimulation/Agents/Environment.m

3.15 <EnvironmentDisplayDelegate> Protocol Reference

`#import <EnvironmentDisplayDelegate.h>`

Inheritance diagram for <EnvironmentDisplayDelegate>:



Public Member Functions

- (void) - addAirplanesToMap:
- (void) - addAirplaneToMap:

- (void) - crashAirplane:
- (void) - landAirplane:
- (void) - updateAirplanesPositions:
- (void) - displayZones:
- (void) - displayZonesControllers:
- (void) - displayAirportControllers:

3.15.1 Detailed Description

Defines a protocol a view controller needs to respond to in order for the environment to create a representation of the simulation.

3.15.2 Member Function Documentation

3.15.2.1 - (void) addAirplanesToMap: dummy(NSArray ∗) *newAirplanes*

Adds a list of airplanes to the map.

Parameters

newAirplanes	The airplanes to add to the interface.

Reimplemented in MainInterfaceController.

3.15.2.2 - (void) addAirplaneToMap: dummy(**Airplane** ∗) *newAirplane*

Add a single airplane to the map.

Parameters

newAirplane	The airplane to add.

Reimplemented in MainInterfaceController.

3.15.2.3 - (void) crashAirplane: dummy(**Airplane** ∗) *airplane*

Crashes an airplane, to inform the user a collision or an accident happened.

Parameters

airplane	

Reimplemented in MainInterfaceController.

3.15.2.4 - (void) displayAirportControllers: dummy(NSDictionary *) *airportsControllers*

Displays the runway on the map.

Parameters

airports-Controllers	

Reimplemented in MainInterfaceController.

3.15.2.5 - (void) displayZones: dummy(NSArray *) *zones*

Prints the borders on the map.

Parameters

zones	An array containing all the zones, each one referencing its segments.

Reimplemented in MainInterfaceController.

3.15.2.6 - (void) displayZonesControllers: dummy(NSDictionary *) *zonesControllers*

Displays the zone controllers on the map.

Parameters

zones-Controllers	

Reimplemented in MainInterfaceController.

3.15.2.7 - (void) landAirplane: dummy(Airplane *) *airplane*

Lands an airplane once it has reached its destination.

Parameters

airplane	

Reimplemented in MainInterfaceController.

3.15.2.8 - (void) updateAirplanesPositions: dummy(NSArray *) *airplanes*

Updates the location of the airplanes on the map.

Parameters

new-Airplanes	

Reimplemented in MainInterfaceController.

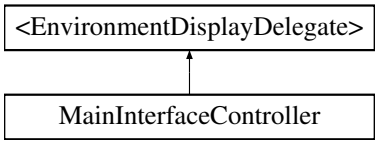
The documentation for this protocol was generated from the following file:

- ATCSimulation/EnvironmentDisplayDelegate.h

3.16 MainInterfaceController Class Reference

`#import <MainInterfaceController.h>`

Inheritance diagram for MainInterfaceController:



Public Member Functions

- (IBAction) - **startStopPressed:**
- (id) - **initWithNibName:bundle:** `[implementation]`
- (void) - **didReceiveMemoryWarning** `[implementation]`
- (void) - **loadView** `[implementation]`
- (void) - **viewDidLoad** `[implementation]`
- (void) - **viewDidUnload** `[implementation]`
- (BOOL) - **shouldAutorotateToInterfaceOrientation:** `[implementation]`
- (void) - addAirplanesToMap: `[implementation]`
- (void) - addAirplaneToMap: `[implementation]`
- (void) - crashAirplane: `[implementation]`
- (void) - landAirplane: `[implementation]`
- (void) - updateAirplanesPositions: `[implementation]`
- (void) - displayZones: `[implementation]`
- (void) - displayZonesControllers: `[implementation]`
- (void) - displayAirportControllers: `[implementation]`
- (void) - **dealloc** `[implementation]`

Properties

- IBOutlet UIButton * **startStopButton**
- IBOutlet MapView * **mapView**
- IBOutlet UIView * **controllersView**
- IBOutlet UIView * **airplanesView**
- int **simulationState**
- Environment * **environment**
- NSMutableDictionary * **airplanesDictionary**

Private Member Functions

- (void) - **createViewsForInterface** `[implementation]`

3.16.1 Detailed Description

The view controller for the interface, communicating with the user and the simulation.

3.16.2 Member Function Documentation

3.16.2.1 - (void) addAirplanesToMap: dummy(NSArray *) *newAirplanes* `[implementation]`

Adds a list of airplanes to the map.

Parameters

newAirplanes	The airplanes to add to the interface.

Reimplemented from <EnvironmentDisplayDelegate>.

3.16.2.2 - (void) addAirplaneToMap: dummy(Airplane *) *newAirplane* `[implementation]`

Add a single airplane to the map.

Parameters

newAirplane	The airplane to add.

Reimplemented from <EnvironmentDisplayDelegate>.

3.16.2.3 - (void) crashAirplane: dummy(Airplane *) airplane [implementation]

Crashes an airplane, to inform the user a collision or an accident happened.

Parameters

airplane	

Reimplemented from <EnvironmentDisplayDelegate>.

3.16.2.4 - (void) displayAirportControllers: dummy(NSDictionary *) airportsControllers [implementation]

Displays the runway on the map.

Parameters

airports-Controllers	

Reimplemented from <EnvironmentDisplayDelegate>.

3.16.2.5 - (void) displayZones: dummy(NSArray *) zones [implementation]

Prints the borders on the map.

Parameters

zones	An array containing all the zones, each one referencing its segments.

Reimplemented from <EnvironmentDisplayDelegate>.

3.16.2.6 - (void) displayZonesControllers: dummy(NSDictionary *) zonesControllers [implementation]

Displays the zone controllers on the map.

Parameters

zones-Controllers	

Reimplemented from <EnvironmentDisplayDelegate>.

3.16.2.7 - (void) landAirplane: dummy(Airplane *) airplane [implementation]

Lands an airplane once it has reached its destination.

Parameters

airplane	

Reimplemented from <EnvironmentDisplayDelegate>.

3.16.2.8 - (void) updateAirplanesPositions: dummy(NSArray *) *airplanes* [implementation]

Updates the location of the airplanes on the map.

Parameters

new-Airplanes	

Reimplemented from <EnvironmentDisplayDelegate>.

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

- ATCSimulation/MainInterfaceController.h
- ATCSimulation/MainInterfaceController.m

3.17 MapView Class Reference

Public Member Functions

- (id) - **initWithFrame:** [implementation]
- (void) - **drawRect:** [implementation]

Properties

- NSArray * **zonesAndTheirBorders**

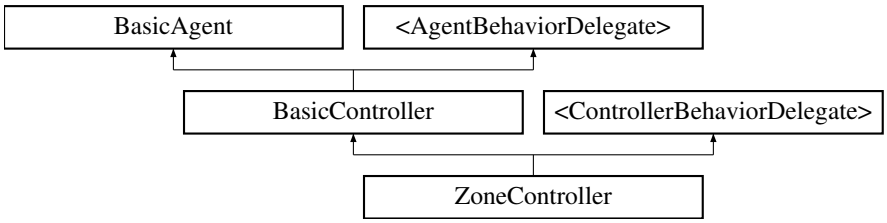
The documentation for this class was generated from the following files:

- ATCSimulation/MapView.h
- ATCSimulation/MapView.m

3.18 ZoneController Class Reference

```
#import <ZoneController.h>
```

Inheritance diagram for ZoneController:



Public Member Functions

- (id) - **init** [implementation]
- (void) - finishMessageAnalysis:withMessageCode:from:originallyTo: [implementation]

Private Member Functions

- (void) - **analyzePosition:fromAirplaneName:** [implementation]

3.18.1 Detailed Description

One of the two specialized agent playing the role of a controller. It can track the airplanes, and communicate with them.

3.18.2 Member Function Documentation

3.18.2.1 - (void) finishMessageAnalysis: dummy(NSString *) *messageContent* withMessageCode:(NVMessageCode) *code* from:(NSString *) *sender* originallyTo:(NSString *) *originalReceiver* [implementation]

This method is usually called if the BasicController was not able to use the message received. It transmits the message calling maybe more specific methods of the controller, with the content of the initial message.

Parameters

messageContent	The content of the message, containing formatted information.
code	The code of the message.
sender	The initial sender of the message.
originalReceiver	The destination of the message, which can this agent, or all the agents etc.

Reimplemented from <ControllerBehaviorDelegate>.

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

- ATCSimulation/Agents/ZoneController.h

- ATCSimulation/Agents/ZoneController.m