

AirDolphin

Generated by Doxygen 1.8.17

1 Data Structure Documentation	1
1.1 airport Struct Reference	1
1.1.1 Detailed Description	1
1.1.2 Field Documentation	1
1.1.2.1 landingQueue	1
1.1.2.2 parkingPlanes	2
1.1.2.3 parkingSize	2
1.1.2.4 planesInRange	2
1.1.2.5 runways	2
1.1.2.6 waitForRunwayQueue	2
1.2 Anchor Struct Reference	2
1.2.1 Detailed Description	3
1.2.2 Field Documentation	3
1.2.2.1 x	3
1.2.2.2 y	3
1.3 button Struct Reference	3
1.3.1 Detailed Description	3
1.3.2 Field Documentation	3
1.3.2.1 action	4
1.3.2.2 center	4
1.3.2.3 CID	4
1.3.2.4 CSG	4
1.3.2.5 selected	4
1.3.2.6 text	4
1.4 chainItem Struct Reference	4
1.4.1 Detailed Description	5
1.4.2 Field Documentation	5
1.4.2.1 data	5
1.4.2.2 index	5
1.4.2.3 next	5
1.4.2.4 previous	5
1.5 list Struct Reference	6
1.5.1 Detailed Description	6
1.5.2 Field Documentation	6
1.5.2.1 comparator	6
1.5.2.2 first	6
1.5.2.3 last	6
1.5.2.4 length	6
1.6 plane Struct Reference	7
1.6.1 Detailed Description	7
1.6.2 Field Documentation	7
1.6.2.1 matriculation	7

1.6.2.2 passengers	7
1.6.2.3 passengersMax	7
1.6.2.4 status	7
1.6.2.5 targetRunway	8
1.6.2.6 type	8
1.7 runway Struct Reference	8
1.7.1 Detailed Description	8
1.7.2 Field Documentation	8
1.7.2.1 id	8
1.7.2.2 length	9
1.7.2.3 maxTakeoffQueue	9
1.7.2.4 planeLT	9
1.7.2.5 takeoffQueue	9
1.7.2.6 type	9
1.7.2.7 width	9
1.8 sim_planeActor Struct Reference	9
1.8.1 Detailed Description	10
1.8.2 Field Documentation	10
1.8.2.1 plane	10
1.8.2.2 stateLengthTimeInMs	10
1.8.2.3 stateRemainTimeInMs	10
1.9 simulation Struct Reference	10
1.9.1 Detailed Description	11
1.9.2 Field Documentation	11
1.9.2.1 airport	11
1.9.2.2 planeActors	11
1.9.2.3 simulationSpeedInMs	11
2 File Documentation	13
2.1 AirManager.c File Reference	13
2.1.1 Detailed Description	14
2.1.2 Function Documentation	14
2.1.2.1 addPlaneToAFRQ()	14
2.1.2.2 addPlaneToLandingQueue()	15
2.1.2.3 addPlaneToParking()	15
2.1.2.4 addPlaneToRunway()	15
2.1.2.5 addPlaneToRunwayQueue()	16
2.1.2.6 buildAirport()	16
2.1.2.7 canAPlaneInLQLandHere()	16
2.1.2.8 canItLandHere()	18
2.1.2.9 grantNextInAFRQAccessToRunway()	18
2.1.2.10 grantNextInLQAccessToRunway()	19

2.1.2.11 grantPlaneInAFRQAccessToRunway()	19
2.1.2.12 grantPlaneInLQAccessToRunway()	19
2.1.2.13 grantTakeoffForRunway()	20
2.1.2.14 isParkingFull()	20
2.1.2.15 isParkingQueueFull()	20
2.1.2.16 isRunwayFree()	21
2.1.2.17 isRunwayQueueFull()	21
2.1.2.18 loadPlaneInAirport()	21
2.1.2.19 newAirport()	22
2.1.2.20 newPlane()	22
2.1.2.21 newRunway()	22
2.1.2.22 planeExitRunway()	23
2.1.2.23 removePlane()	23
2.2 AirManager.h File Reference	24
2.2.1 Detailed Description	25
2.2.2 Enumeration Type Documentation	25
2.2.2.1 planeStatus	25
2.2.2.2 planeType	26
2.2.2.3 runwayType	26
2.2.3 Function Documentation	26
2.2.3.1 addPlaneToAFRQ()	26
2.2.3.2 addPlaneToLandingQueue()	28
2.2.3.3 addPlaneToParking()	28
2.2.3.4 addPlaneToRunway()	28
2.2.3.5 addPlaneToRunwayQueue()	29
2.2.3.6 buildAirport()	29
2.2.3.7 canAPlaneInLQLandHere()	29
2.2.3.8 canItLandHere()	31
2.2.3.9 grantNextInAFRQAccessToRunway()	31
2.2.3.10 grantNextInLQAccessToRunway()	32
2.2.3.11 grantPlaneInAFRQAccessToRunway()	32
2.2.3.12 grantPlaneInLQAccessToRunway()	32
2.2.3.13 grantTakeoffForRunway()	33
2.2.3.14 isParkingFull()	33
2.2.3.15 isParkingQueueFull()	33
2.2.3.16 isRunwayFree()	34
2.2.3.17 isRunwayQueueFull()	34
2.2.3.18 loadPlaneInAirport()	34
2.2.3.19 newAirport()	35
2.2.3.20 newPlane()	35
2.2.3.21 newRunway()	35
2.2.3.22 planeExitRunway()	36

2.2.3.23 removePlane()	36
2.3 AirSim.c File Reference	37
2.3.1 Detailed Description	37
2.3.2 Function Documentation	37
2.3.2.1 cmprPointer()	37
2.3.2.2 getSimPlaneActorInList()	38
2.3.2.3 initSimulation()	38
2.3.2.4 main()	39
2.3.2.5 msleep()	39
2.3.2.6 newSimPlaneActor()	39
2.3.2.7 planeNextAction()	40
2.4 AirSim.h File Reference	40
2.4.1 Detailed Description	40
2.4.2 Function Documentation	41
2.4.2.1 getSimPlaneActorInList()	41
2.5 bddManager.c File Reference	41
2.5.1 Detailed Description	42
2.5.2 Function Documentation	42
2.5.2.1 CmpPtr()	42
2.5.2.2 openChainFile()	42
2.5.2.3 randomInt()	43
2.5.2.4 randomRegistration()	43
2.5.2.5 savePlanesInFile()	43
2.5.2.6 sMakeChainData()	44
2.6 bddManager.h File Reference	44
2.6.1 Detailed Description	44
2.6.2 Function Documentation	44
2.6.2.1 openChainFile()	45
2.6.2.2 randomInt()	46
2.6.2.3 randomRegistration()	46
2.6.2.4 savePlanesInFile()	46
2.7 cmdPrint.c File Reference	47
2.7.1 Detailed Description	47
2.7.2 Function Documentation	47
2.7.2.1 debugPlaneStatus()	47
2.7.2.2 debugPlaneType()	48
2.7.2.3 debugPrintAirport()	48
2.7.2.4 debugPrintPlane()	48
2.7.2.5 debugPrintRunway()	49
2.7.2.6 debugRunwayType()	49
2.7.2.7 printParkingsList()	49
2.7.2.8 printPlanesList()	49

2.7.2.9 printRunwaysList()	50
2.8 cmdPrint.h File Reference	50
2.8.1 Detailed Description	51
2.8.2 Function Documentation	51
2.8.2.1 debugPrintAirport()	51
2.8.2.2 debugPrintPlane()	51
2.8.2.3 debugPrintRunway()	52
2.9 Renderer.c File Reference	52
2.9.1 Detailed Description	54
2.9.2 Enumeration Type Documentation	54
2.9.2.1 menuAction	54
2.9.2.2 textAlign	54
2.9.3 Function Documentation	54
2.9.3.1 getActionButton()	55
2.9.3.2 initWindow()	55
2.9.3.3 interf_AirportToRender()	55
2.9.3.4 interf_launchMenu()	56
2.9.3.5 interf_Parking()	56
2.9.3.6 interf_Parking_PrintLine()	56
2.9.3.7 interf_Radar()	56
2.9.3.8 interf_Radar_PrintLine()	57
2.9.3.9 interf_Runway()	57
2.9.3.10 isButtonHover()	58
2.9.3.11 newButton()	58
2.9.3.12 printButtons()	58
2.9.3.13 printProgress()	59
2.9.3.14 printRectangleWithBorder()	59
2.9.3.15 printText()	60
2.9.3.16 SetDrawColor()	60
2.9.3.17 updateAirportRenderer()	60
2.9.3.18 updateHoverButtons()	61
2.10 Renderer.h File Reference	61
2.10.1 Detailed Description	61
2.10.2 Function Documentation	61
2.10.2.1 initWindow()	61
2.10.2.2 updateAirportRenderer()	62
2.11 SmartList.c File Reference	62
2.11.1 Detailed Description	63
2.11.2 Function Documentation	63
2.11.2.1 appendAtInList()	63
2.11.2.2 appendInList()	63
2.11.2.3 deleteInList()	64

2.11.2.4 deleteItemAtIndex()	64
2.11.2.5 emptyList()	64
2.11.2.6 getDataAtIndex()	65
2.11.2.7 getItemAtIndex()	65
2.11.2.8 newChainItem()	65
2.11.2.9 newList()	66
2.11.2.10 pushInList()	66
2.11.2.11 putItemAtIndex()	66
2.11.2.12 searchDataInList()	68
2.11.2.13 searchIndexInList()	68
2.11.2.14 updateIndexs()	68
2.12 SmartList.h File Reference	69
2.12.1 Detailed Description	70
2.12.2 Function Documentation	70
2.12.2.1 appendAtInList()	70
2.12.2.2 appendInList()	70
2.12.2.3 deleteInList()	70
2.12.2.4 deleteItemAtIndex()	71
2.12.2.5 emptyList()	71
2.12.2.6 getDataAtIndex()	71
2.12.2.7 getItemAtIndex()	72
2.12.2.8 newList()	72
2.12.2.9 pushInList()	73
2.12.2.10 searchDataInList()	73
2.12.2.11 searchIndexInList()	73

Chapter 1

Data Structure Documentation

1.1 airport Struct Reference

Airport containing all the planes and runways of the simulation.

```
#include <AirManager.h>
```

Data Fields

- unsigned int [parkingSize](#)
- [list](#) * [runways](#)
- [list](#) * [planesInRange](#)
- [list](#) * [parkingPlanes](#)
- [list](#) * [landingQueue](#)
- [list](#) * [waitForRunwayQueue](#)

1.1.1 Detailed Description

Airport containing all the planes and runways of the simulation.

1.1.2 Field Documentation

1.1.2.1 landingQueue

```
list* landingQueue
```

All planes in the landing queue

1.1.2.2 parkingPlanes

`list*` parkingPlanes

Planes in the parking

1.1.2.3 parkingSize

`unsigned int` parkingSize

Size of the parking

1.1.2.4 planesInRange

`list*` planesInRange

All the planes in range of the control tower registration system

1.1.2.5 runways

`list*` runways

Runways of Airport

1.1.2.6 waitForRunwayQueue

`list*` waitForRunwayQueue

Planes in parking waiting for a landing

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

- [AirManager.h](#)

1.2 Anchor Struct Reference

2D Coordinates set

Data Fields

- `int` `x`
- `int` `y`

1.2.1 Detailed Description

2D Coordinates set

1.2.2 Field Documentation

1.2.2.1 x

```
int x
```

x coordinates

1.2.2.2 y

```
int y
```

y coordinates

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

- [Renderer.c](#)

1.3 button Struct Reference

Button item for rendering.

Data Fields

- [Anchor CSG](#)
- [Anchor CID](#)
- [Anchor center](#)
- [menuAction action](#)
- bool [selected](#)
- char * [text](#)

1.3.1 Detailed Description

Button item for rendering.

1.3.2 Field Documentation

1.3.2.1 action

`menuAction` action

Action of the button

1.3.2.2 center

`Anchor` center

Center point

1.3.2.3 CID

`Anchor` CID

Inferior right point

1.3.2.4 CSG

`Anchor` CSG

Top left point

1.3.2.5 selected

`bool` selected

Is the button selected

1.3.2.6 text

`char*` text

Text on the button

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

- [Renderer.c](#)

1.4 chainItem Struct Reference

Item container of list.

```
#include <SmartList.h>
```

Data Fields

- void * [data](#)
- int [index](#)
- [chainItem](#) * [previous](#)
- [chainItem](#) * [next](#)

1.4.1 Detailed Description

Item container of list.

1.4.2 Field Documentation

1.4.2.1 data

```
void* data
```

Pointer to stored data

1.4.2.2 index

```
int index
```

Index of item

1.4.2.3 next

```
chainItem* next
```

Next item in list, NULL if none

1.4.2.4 previous

```
chainItem* previous
```

Previous item in list, NULL if none

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

- [SmartList.h](#)

1.5 list Struct Reference

List object.

```
#include <SmartList.h>
```

Data Fields

- int [length](#)
- [chainItem](#) * [first](#)
- [chainItem](#) * [last](#)
- compareTwoPointersFunction [comparator](#)

1.5.1 Detailed Description

List object.

1.5.2 Field Documentation

1.5.2.1 comparator

```
compareTwoPointersFunction comparator
```

Function to compare data on the list

1.5.2.2 first

```
chainItem* first
```

First item of the list

1.5.2.3 last

```
chainItem* last
```

Last item of the list

1.5.2.4 length

```
int length
```

Length of list

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

- [SmartList.h](#)

1.6 plane Struct Reference

Plane status and datas.

```
#include <AirManager.h>
```

Data Fields

- char [matriculation](#) [7]
- [planeType](#) type
- unsigned int [passengers](#)
- unsigned int [passengersMax](#)
- [planeStatus](#) status
- void * [targetRunway](#)

1.6.1 Detailed Description

Plane status and datas.

1.6.2 Field Documentation

1.6.2.1 matriculation

```
char matriculation[7]
```

Normalized matriculation of the planes

1.6.2.2 passengers

```
unsigned int passengers
```

Number of passengers

1.6.2.3 passengersMax

```
unsigned int passengersMax
```

Number of max passengers

1.6.2.4 status

```
planeStatus status
```

Status of the plane

1.6.2.5 targetRunway

```
void* targetRunway
```

Current runway of the plane, NULL if not on a runway

1.6.2.6 type

```
planeType type
```

Class of plane

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

- [AirManager.h](#)

1.7 runway Struct Reference

Runway of an airport.

```
#include <AirManager.h>
```

Data Fields

- char [id](#)
- float [length](#)
- float [width](#)
- [runwayType](#) type
- unsigned int [maxTakeoffQueue](#)
- list * [takeoffQueue](#)
- plane * [planeLT](#)

1.7.1 Detailed Description

Runway of an airport.

1.7.2 Field Documentation

1.7.2.1 id

```
char id
```

Runway identifier

1.7.2.2 length

```
float length
```

Length of runway in meters

1.7.2.3 maxTakeoffQueue

```
unsigned int maxTakeoffQueue
```

Size of the take off queue

1.7.2.4 planeLT

```
plane* planeLT
```

Plane currently landing or taking off on the runway

1.7.2.5 takeoffQueue

```
list* takeoffQueue
```

List of planes in take off queue

1.7.2.6 type

```
runwayType type
```

Runway class

1.7.2.7 width

```
float width
```

Width of runway in meters

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

- [AirManager.h](#)

1.8 sim_planeActor Struct Reference

Sstructure of plane Actor.

```
#include <AirSim.h>
```

Data Fields

- [plane * plane](#)
- int [stateRemainTimeInMs](#)
- int [stateLengthTimeInMs](#)

1.8.1 Detailed Description

Sstructure of plane Actor.

1.8.2 Field Documentation

1.8.2.1 plane

[plane*](#) [plane](#)

Plane of planeActor

1.8.2.2 stateLengthTimeInMs

int [stateLengthTimeInMs](#)

State remaining length time for the current task in milliseconds

1.8.2.3 stateRemainTimeInMs

int [stateRemainTimeInMs](#)

State remaining time for the current task in milliseconds

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

- [AirSim.h](#)

1.9 simulation Struct Reference

Structure of simulation.

```
#include <AirSim.h>
```

Data Fields

- unsigned int [simulationSpeedInMs](#)
- [airport](#) * [airport](#)
- [list](#) * [planeActors](#)

1.9.1 Detailed Description

Structure of simulation.

1.9.2 Field Documentation

1.9.2.1 airport

```
airport* airport
```

Airport of the simulation

1.9.2.2 planeActors

```
list* planeActors
```

All of planes actors

1.9.2.3 simulationSpeedInMs

```
unsigned int simulationSpeedInMs
```

Simulation speed in milleseconds

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

- [AirSim.h](#)

Chapter 2

File Documentation

2.1 AirManager.c File Reference

Functions for the Manager.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "AirManager.h"
```

Functions

- int **comparePointer** (void *data1, void *data2)
- plane * **newPlane** (char matriculation[7], planeType type, unsigned int passengers, unsigned int passengersMax, planeStatus status)
Create and allocate memory for a plane.
- void **removePlane** (airport *airport, plane *plane)
Remove a plane from Airport.
- void **loadPlaneInAirport** (airport *airport, plane *plane)
Add a plane to a specified Airport.
- bool **canItLandHere** (plane *plane, runway *runway)
Determine if a plane could land on the specified runway, based on his size and the class of the runway.
- bool **canAPlaneInLQLandHere** (airport *airport, runway *runway)
*Determine if a plane currently in Landing Queue can land on the specified runway.
This function take into account the size of the plane, the class of the runway and if the parking is full or is going to be full.*
- runway * **newRunway** (float length, float width, runwayType type, unsigned int maxTakeoffQueue)
Create and allocate memory for a runway.
- bool **isRunwayFree** (runway *runway)
Determine if a runway is curently used by a plane or not.
- void **addPlaneToRunway** (runway *runway, plane *plane)
*Add Plane to the runway, updating it status.
Used to allow an airplane to land or takeoff on a runway.*
- void **planeExitRunway** (runway *runway, plane *plane)

- Remove plane from runway, updating it status.*
Used to finish the action started on runway, either landing or taking off.

 - void `grantTakeoffForRunway` (`runway *runway`)
Allow next in runway queue for take off.
 - void `addPlaneToRunwayQueue` (`runway *runway`, `plane *plane`)
Add plane in the take off queue for a specified runway.
 - bool `isRunwayQueueFull` (`runway *runway`)
Determine if the take off queue of a runway is full.
 - `airport * newAirport` (unsigned int parkingSize)
Create an aiport and allocate adequate memory.
 - void `buildAirport` (`airport *airport`, int numberOfSmallRunway, int numberOfMediumRunway, int numberOfLargeRunway)
Build the runways for an airport.
 - void `addPlaneToParking` (`airport *airport`, `plane *plane`)
Add plane to Parking.
 - bool `isParkingFull` (`airport *airport`)
Determine if the parking is full.
 - bool `isParkingQueueFull` (`airport *airport`)
Determine if the parking is full or if the planes currently landing will fill the parking.
 - void `addPlaneToLandingQueue` (`airport *airport`, `plane *plane`)
Add plane to landing queue.
 - void `grantPlaneInLQAccessToRunway` (`airport *airport`, `runway *runway`, `plane *plane`)
Give to a plane in landing queue access to specified runway.
 - void `grantNextInLQAccessToRunway` (`airport *airport`, `runway *runway`)
Grant next plane in landing queue access to runway.
 - void `addPlaneToAFRQ` (`airport *airport`, `plane *plane`)
Add plane to the queue for acces to runway.
 - void `grantPlaneInAFRQAccessToRunway` (`airport *airport`, `runway *runway`, `plane *plane`)
Give to a plane in the queue for takeoff runway access to specified runway queue.
 - void `grantNextInAFRQAccessToRunway` (`airport *airport`, `runway *runway`)
Grant the next plane waiting for acces to runway access to runway.

2.1.1 Detailed Description

Functions for the Manager.

2.1.2 Function Documentation

2.1.2.1 addPlaneToAFRQ()

```
void addPlaneToAFRQ (
    airport * airport,
    plane * plane )
```

Add plane to the queue for acces to runway.

Parameters

<i>airport</i>	
<i>plane</i>	

2.1.2.2 addPlaneToLandingQueue()

```
void addPlaneToLandingQueue (
    airport * airport,
    plane * plane )
```

Add plane to landing queue.

Parameters

<i>airport</i>	
<i>plane</i>	

2.1.2.3 addPlaneToParking()

```
void addPlaneToParking (
    airport * airport,
    plane * plane )
```

Add plane to Parking.

Parameters

in	<i>airport</i>	
in	<i>plane</i>	

2.1.2.4 addPlaneToRunway()

```
void addPlaneToRunway (
    runway * runway,
    plane * plane )
```

Add Plane to the runway, updating it status.
Used to allow an airplane to land or takeoff on a runway.

Parameters

in	<i>runway</i>	Runway where the plane should take place on.
in	<i>plane</i>	The plane.

2.1.2.5 addPlaneToRunwayQueue()

```
void addPlaneToRunwayQueue (
    runway * runway,
    plane * plane )
```

Add plane in the take off queue for a specified runway.

Parameters

in	<i>runway</i>	
in	<i>plane</i>	

2.1.2.6 buildAirport()

```
void buildAirport (
    airport * airport,
    int numberOfSmallRunway,
    int numberOfMediumRunway,
    int numberOfLargeRunway )
```

Build the runways for an airport.

Parameters

in	<i>airport</i>	
in	<i>numberOfSmallRunway</i>	
in	<i>numberOfMediumRunway</i>	
in	<i>numberOfLargeRunway</i>	

2.1.2.7 canAPlaneInLQLandHere()

```
bool canAPlaneInLQLandHere (
    airport * airport,
    runway * runway )
```


Determine if a plane currently in Landing Queue can land on the specified runway.

This function take into account the size of the plane, the class of the runway and if the parking is full or is going to be full.

Parameters

in	<i>airport</i>	Airport containing the Landing Queue
in	<i>runway</i>	Runway to test

Returns

true if one of the planes in the landing queue can land on the runway
false if no plane in LQ can land on the runway

2.1.2.8 canItLandHere()

```
bool canItLandHere (
    plane * plane,
    runway * runway )
```

Determine if a plane could land on the specified runway, based on his size and the class of the runway.

Parameters

in	<i>plane</i>	
in	<i>runway</i>	

Returns

true if the plane could land on the runway
false if he can't

2.1.2.9 grantNextInAFRQAccessToRunway()

```
void grantNextInAFRQAccessToRunway (
    airport * airport,
    runway * runway )
```

Grant the next plane waiting for acces to runway access to runway.

Parameters

<i>airport</i>	
<i>runway</i>	

2.1.2.10 grantNextInLQAccessToRunway()

```
void grantNextInLQAccessToRunway (
    airport * airport,
    runway * runway )
```

Grant next plane in landing queue access to runway.

Parameters

<i>airport</i>	
<i>runway</i>	

2.1.2.11 grantPlaneInAFRQAccessToRunway()

```
void grantPlaneInAFRQAccessToRunway (
    airport * airport,
    runway * runway,
    plane * plane )
```

Give to a plane in the queue for takeoff runway access to specified runway queue.

Parameters

in	<i>airport</i>	
in	<i>runway</i>	
in	<i>plane</i>	

2.1.2.12 grantPlaneInLQAccessToRunway()

```
void grantPlaneInLQAccessToRunway (
    airport * airport,
    runway * runway,
    plane * plane )
```

Give to a plane in landing queue access to specified runway.

Parameters

in	<i>airport</i>	
in	<i>runway</i>	
in	<i>plane</i>	

2.1.2.13 grantTakeoffForRunway()

```
void grantTakeoffForRunway (
    runway * runway )
```

Allow next in runway queue for take off.

Parameters

in	<i>runway</i>	
----	---------------	--

2.1.2.14 isParkingFull()

```
bool isParkingFull (
    airport * airport )
```

Determine if the parking is full.

Parameters

<i>airport</i>	containg the parking
----------------	----------------------

Returns

true if the parking is full
false if the parking is not full

2.1.2.15 isParkingQueueFull()

```
bool isParkingQueueFull (
    airport * airport )
```

Determine if the parking is full or if the planes currently landing will fill the parking.

Parameters

<i>airport</i>	
----------------	--

Returns

true if the parking is or will be full
false if it isn't

2.1.2.16 isRunwayFree()

```
bool isRunwayFree (
    runway * runway )
```

Determine if a runway is currently used by a plane or not.

Parameters

in	<i>runway</i>	
----	---------------	--

Returns

true if the runway is empty

false if the runway is currently used

2.1.2.17 isRunwayQueueFull()

```
bool isRunwayQueueFull (
    runway * runway )
```

Determine if the take off queue of a runway is full.

Parameters

in	<i>runway</i>	
----	---------------	--

Returns

true if the take off queue is full

false if the take off queue is not full

2.1.2.18 loadPlaneInAirport()

```
void loadPlaneInAirport (
    airport * airport,
    plane * plane )
```

Add a plane to a specified Airport.

Parameters

in	<i>airport</i>	Airport to load plane in
in	<i>plane</i>	Plane to import

2.1.2.19 newAirport()

```
airport* newAirport (
    unsigned int parkingSize )
```

Create an airport and allocate adequate memory.

Parameters

in	<i>parkingSize</i>	Size of the parking
----	--------------------	---------------------

Returns

airport pointer to the new airport

2.1.2.20 newPlane()

```
plane* newPlane (
    char matriculation[7],
    planeType type,
    unsigned int passengers,
    unsigned int passengersMax,
    planeStatus status )
```

Create and allocate memory for a plane.

Parameters

in	<i>matriculation</i>	Matriculation of the plane
in	<i>type</i>	Class of the plane (AIRLINER, LIGHT or BUSINESS)
in	<i>passengers</i>	Number of passengers currently on board
in	<i>passengersMax</i>	Maximum number of passengers
in	<i>status</i>	Status of the plane (FLYING or PARKING only)

Returns

Pointer the created plane

2.1.2.21 newRunway()

```
runway* newRunway (
    float length,
```

```
float width,  
runwayType type,  
unsigned int maxTakeoffQueue )
```

Create and allocate memory for a runway.

Parameters

in	<i>length</i>	Length of runway in meters.
in	<i>width</i>	Width of runway in meters.
in	<i>type</i>	Class of runway (SMALL, MEDIUM, LARGE).
in	<i>maxTakeoffQueue</i>	Size of take off queue.

Returns

Pointer to the created plane.

2.1.2.22 planeExitRunway()

```
void planeExitRunway (  
    runway * runway,  
    plane * plane )
```

Remove plane from runway, updating it status.
Used to finish the action started on runway, either landing or taking off.

Parameters

in	<i>runway</i>	Runway from wich the plane exit.
in	<i>plane</i>	The plane.

2.1.2.23 removePlane()

```
void removePlane (  
    airport * airport,  
    plane * plane )
```

Remove a plane from Airport.

Parameters

in	<i>airport</i>	Airport from where the plane should be removed
in	<i>plane</i>	Plane to remove

2.2 AirManager.h File Reference

Manage planes, runways and airports status and event based behaviours.

```
#include <stdbool.h>
#include "SmartList.h"
```

Data Structures

- struct [airport](#)
Airport containing all the planes and runways of the simulation.
- struct [plane](#)
Plane status and datas.
- struct [runway](#)
Runway of an airport.

Enumerations

- enum [planeStatus](#) {
FLYING, WAITING_LANDING, LANDING, PARKING,
WAITING_TAKEOFF, TAKEOFF }
Enumeration of planes status.
- enum [planeType](#) { AIRLINER, BUSINESS, LIGHT }
Enumeration of the planed types.
- enum [runwayType](#) { SMALL, MEDIUM, LARGE }
Enumeration of runways classes.

Functions

- [plane](#) * [newPlane](#) (char matriculation[7], [planeType](#) type, unsigned int passengers, unsigned int passengersMax, [planeStatus](#) status)
Create and allocate memory for a plane.
- void [loadPlaneInAirport](#) ([airport](#) *airport, [plane](#) *plane)
Add a plane to a specified Airport.
- void [removePlane](#) ([airport](#) *airport, [plane](#) *plane)
Remove a plane from Airport.
- bool [canItLandHere](#) ([plane](#) *plane, [runway](#) *runway)
Determine if a plane could land on the specified runway, based on his size and the class of the runway.
- bool [canAPlaneInLQLandHere](#) ([airport](#) *airport, [runway](#) *runway)
*Determine if a plane currently in Landing Queue can land on the specified runway.
This function take into account the size of the plane, the class of the runway and if the parking is full or is going to be full.*
- [runway](#) * [newRunway](#) (float length, float width, [runwayType](#) type, unsigned int maxTakeoffQueue)
Create and allocate memory for a runway.
- void [addPlaneToRunway](#) ([runway](#) *runway, [plane](#) *plane)
*Add Plane to the runway, updating it status.
Used to allow an airplane to land or takeoff on a runway.*
- void [planeExitRunway](#) ([runway](#) *runway, [plane](#) *plane)

- Remove plane from runway, updating it status.*
Used to finish the action started on runway, either landing or taking off.
- bool `isRunwayFree` (`runway *runway`)
Determine if a runway is currently used by a plane or not.
- void `grantTakeoffForRunway` (`runway *runway`)
Allow next in runway queue for take off.
- void `addPlaneToRunwayQueue` (`runway *runway`, `plane *plane`)
Add plane in the take off queue for a specified runway.
- bool `isRunwayQueueFull` (`runway *runway`)
Determine if the take off queue of a runway is full.
- `airport * newAirport` (unsigned int parkingSize)
Create an airport and allocate adequate memory.
- void `buildAirport` (`airport *airport`, int numberOfSmallRunway, int numberOfMediumRunway, int numberOfLargeRunway)
Build the runways for an airport.
- void `addPlaneToParking` (`airport *airport`, `plane *plane`)
Add plane to Parking.
- bool `isParkingFull` (`airport *airport`)
Determine if the parking is full.
- bool `isParkingQueueFull` (`airport *airport`)
Determine if the parking is full or if the planes currently landing will fill the parking.
- void `addPlaneToLandingQueue` (`airport *airport`, `plane *plane`)
Add plane to landing queue.
- void `grantNextInLQAccessToRunway` (`airport *airport`, `runway *runway`)
Grant next plane in landing queue access to runway.
- void `addPlaneToAFRQ` (`airport *airport`, `plane *plane`)
Add plane to the queue for access to runway.
- void `grantNextInAFRQAccessToRunway` (`airport *airport`, `runway *runway`)
Grant the next plane waiting for access to runway access to runway.
- void `grantPlaneInLQAccessToRunway` (`airport *airport`, `runway *runway`, `plane *plane`)
Give to a plane in landing queue access to specified runway.
- void `grantPlaneInAFRQAccessToRunway` (`airport *airport`, `runway *runway`, `plane *plane`)
Give to a plane in the queue for takeoff runway access to specified runway queue.

2.2.1 Detailed Description

Manage planes, runways and airports status and event based behaviours.

2.2.2 Enumeration Type Documentation

2.2.2.1 planeStatus

```
enum planeStatus
```

Enumeration of planes status.

Enumerator

FLYING	Plane is in standby, flying
WAITING_LANDING	Plane is flying and in landing queue
LANDING	Plane is currently landing on a runway
PARKING	Plane is currently at parking
WAITING_TAKEOFF	Plane is in a runway queue
TAKEOFF	Plane is currently taking off on a runway

2.2.2.2 planeType

```
enum planeType
```

Enumeration of the planed types.

Enumerator

AIRLINER	Plane is an Airliner and can land on LARGE runways
BUSINESS	Plane is a Business Class and can land on all runways
LIGHT	Plane is Ligth Class and can land on SMALL runways

2.2.2.3 runwayType

```
enum runwayType
```

Enumeration of runways classes.

Enumerator

SMALL	Small runway
MEDIUM	Medium runway
LARGE	Large runway

2.2.3 Function Documentation**2.2.3.1 addPlaneToAFRQ()**

```
void addPlaneToAFRQ (
    airport * airport,
    plane * plane )
```

Add plane to the queue for acces to runway.

Parameters

<i>airport</i>	
<i>plane</i>	

2.2.3.2 addPlaneToLandingQueue()

```
void addPlaneToLandingQueue (
    airport * airport,
    plane * plane )
```

Add plane to landing queue.

Parameters

<i>airport</i>	
<i>plane</i>	

2.2.3.3 addPlaneToParking()

```
void addPlaneToParking (
    airport * airport,
    plane * plane )
```

Add plane to Parking.

Parameters

in	<i>airport</i>	
in	<i>plane</i>	

2.2.3.4 addPlaneToRunway()

```
void addPlaneToRunway (
    runway * runway,
    plane * plane )
```

Add Plane to the runway, updating it status.
Used to allow an airplane to land or takeoff on a runway.

Parameters

in	<i>runway</i>	Runway where the plane should take place on.
in	<i>plane</i>	The plane.

2.2.3.5 addPlaneToRunwayQueue()

```
void addPlaneToRunwayQueue (
    runway * runway,
    plane * plane )
```

Add plane in the take off queue for a specified runway.

Parameters

in	<i>runway</i>	
in	<i>plane</i>	

2.2.3.6 buildAirport()

```
void buildAirport (
    airport * airport,
    int numberOfSmallRunway,
    int numberOfMediumRunway,
    int numberOfLargeRunway )
```

Build the runways for an airport.

Parameters

in	<i>airport</i>	
in	<i>numberOfSmallRunway</i>	
in	<i>numberOfMediumRunway</i>	
in	<i>numberOfLargeRunway</i>	

2.2.3.7 canAPlaneInLQLandHere()

```
bool canAPlaneInLQLandHere (
    airport * airport,
    runway * runway )
```

Determine if a plane currently in Landing Queue can land on the specified runway.

This function take into account the size of the plane, the class of the runway and if the parking is full or is going to be full.

Parameters

in	<i>airport</i>	Airport containing the Landing Queue
in	<i>runway</i>	Runway to test

Returns

true if one of the planes in the landing queue can land on the runway
false if no plane in LQ can land on the runway

2.2.3.8 canItLandHere()

```
bool canItLandHere (
    plane * plane,
    runway * runway )
```

Determine if a plane could land on the specified runway, based on his size and the class of the runway.

Parameters

in	<i>plane</i>	
in	<i>runway</i>	

Returns

true if the plane could land on the runway
false if he can't

2.2.3.9 grantNextInAFRQAccessToRunway()

```
void grantNextInAFRQAccessToRunway (
    airport * airport,
    runway * runway )
```

Grant the next plane waiting for acces to runway access to runway.

Parameters

<i>airport</i>	
<i>runway</i>	

2.2.3.10 grantNextInLQAccessToRunway()

```
void grantNextInLQAccessToRunway (
    airport * airport,
    runway * runway )
```

Grant next plane in landing queue access to runway.

Parameters

<i>airport</i>	
<i>runway</i>	

2.2.3.11 grantPlaneInAFRQAccessToRunway()

```
void grantPlaneInAFRQAccessToRunway (
    airport * airport,
    runway * runway,
    plane * plane )
```

Give to a plane in the queue for takeoff runway access to specified runway queue.

Parameters

in	<i>airport</i>	
in	<i>runway</i>	
in	<i>plane</i>	

2.2.3.12 grantPlaneInLQAccessToRunway()

```
void grantPlaneInLQAccessToRunway (
    airport * airport,
    runway * runway,
    plane * plane )
```

Give to a plane in landing queue access to specified runway.

Parameters

in	<i>airport</i>	
in	<i>runway</i>	
in	<i>plane</i>	

2.2.3.13 grantTakeoffForRunway()

```
void grantTakeoffForRunway (
    runway * runway )
```

Allow next in runway queue for take off.

Parameters

in	<i>runway</i>	
----	---------------	--

2.2.3.14 isParkingFull()

```
bool isParkingFull (
    airport * airport )
```

Determine if the parking is full.

Parameters

<i>airport</i>	containg the parking
----------------	----------------------

Returns

true if the parking is full
false if the parking is not full

2.2.3.15 isParkingQueueFull()

```
bool isParkingQueueFull (
    airport * airport )
```

Determine if the parking is full or if the planes currently landing will fill the parking.

Parameters

<i>airport</i>	
----------------	--

Returns

true if the parking is or will be full
false if it isn't

2.2.3.16 isRunwayFree()

```
bool isRunwayFree (  
    runway * runway )
```

Determine if a runway is currently used by a plane or not.

Parameters

in	<i>runway</i>	
----	---------------	--

Returns

true if the runway is empty
false if the runway is currently used

2.2.3.17 isRunwayQueueFull()

```
bool isRunwayQueueFull (  
    runway * runway )
```

Determine if the take off queue of a runway is full.

Parameters

in	<i>runway</i>	
----	---------------	--

Returns

true if the take off queue is full
false if the take off queue is not full

2.2.3.18 loadPlaneInAirport()

```
void loadPlaneInAirport (  
    airport * airport,  
    plane * plane )
```

Add a plane to a specified Airport.

Parameters

in	<i>airport</i>	Airport to load plane in
in	<i>plane</i>	Plane to import

2.2.3.19 newAirport()

```
airport* newAirport (
    unsigned int parkingSize )
```

Create an airport and allocate adequate memory.

Parameters

in	<i>parkingSize</i>	Size of the parking
----	--------------------	---------------------

Returns

airport pointer to the new airport

2.2.3.20 newPlane()

```
plane* newPlane (
    char matriculation[7],
    planeType type,
    unsigned int passengers,
    unsigned int passengersMax,
    planeStatus status )
```

Create and allocate memory for a plane.

Parameters

in	<i>matriculation</i>	Matriculation of the plane
in	<i>type</i>	Class of the plane (AIRLINER, LIGHT or BUSINESS)
in	<i>passengers</i>	Number of passengers currently on board
in	<i>passengersMax</i>	Maximum number of passengers
in	<i>status</i>	Status of the plane (FLYING or PARKING only)

Returns

Pointer the created plane

2.2.3.21 newRunway()

```
runway* newRunway (
    float length,
```

```
float width,
runwayType type,
unsigned int maxTakeoffQueue )
```

Create and allocate memory for a runway.

Parameters

in	<i>length</i>	Length of runway in meters.
in	<i>width</i>	Width of runway in meters.
in	<i>type</i>	Class of runway (SMALL, MEDIUM, LARGE).
in	<i>maxTakeoffQueue</i>	Size of take off queue.

Returns

Pointer to the created plane.

2.2.3.22 planeExitRunway()

```
void planeExitRunway (
    runway * runway,
    plane * plane )
```

Remove plane from runway, updating it status.
Used to finish the action started on runway, either landing or taking off.

Parameters

in	<i>runway</i>	Runway from wich the plane exit.
in	<i>plane</i>	The plane.

2.2.3.23 removePlane()

```
void removePlane (
    airport * airport,
    plane * plane )
```

Remove a plane from Airport.

Parameters

in	<i>airport</i>	Airport from where the plane should be removed
in	<i>plane</i>	Plane to remove

2.3 AirSim.c File Reference

Manage the simulation of planes dynamic behaviours.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "AirSim.h"
#include "Renderer.h"
#include "bddManager.h"
```

Functions

- int [cmprPointer](#) (void *data1, void *data2)
Comparaison of pointers.
- [sim_planeActor](#) * [newSimPlaneActor](#) (char matriculation[7], [planeType](#) type, unsigned int passengers, unsigned int passengersMax, [planeStatus](#) status)
Create a new plane for the simulation.
- [sim_planeActor](#) * [getSimPlaneActorInList](#) ([plane](#) *plane, [simulation](#) *simulation)
Get the Sim Plane Actor In List object.
- [simulation](#) [initSimulation](#) (int parkingSize, int numberOfSmallRunway, int numberOfMediumRunway, int numberOfLargeRunway, int numberOfPlanes)
Initialise the simulation.
- void [planeNextAction](#) ([airport](#) *airport, [sim_planeActor](#) *planeActor)
Give action to the next plane.
- int [msleep](#) (unsigned int tms)
Function for time.
- int [main](#) ()
Function main.

2.3.1 Detailed Description

Manage the simulation of planes dynamic behaviours.

2.3.2 Function Documentation

2.3.2.1 cmprPointer()

```
int cmprPointer (
    void * data1,
    void * data2 )
```

Comparaison of pointers.

Parameters

<i>data1</i>	
<i>data2</i>	

Returns

int

2.3.2.2 getSimPlaneActorInList()

```
sim_planeActor* getSimPlaneActorInList (
    plane * plane,
    simulation * simulation )
```

Get the Sim Plane Actor In List object.

Parameters

<i>plane</i>	
<i>simulation</i>	

Returns

sim_planeActor*

2.3.2.3 initSimulation()

```
simulation initSimulation (
    int parkingSize,
    int numberOfSmallRunway,
    int numberOfMediumRunway,
    int numberOfLargeRunway,
    int numberOfPlanes )
```

Initialise the simulation.

Parameters

<i>parkingSize</i>	
<i>numberOfSmallRunway</i>	
<i>numberOfMediumRunway</i>	
<i>numberOfLargeRunway</i>	
<i>numberOfPlanes</i>	

Returns

simulation

2.3.2.4 main()

```
int main ( )
```

Function main.

Returns

int

2.3.2.5 msleep()

```
int msleep (
    unsigned int tms )
```

Function for time.

Parameters

<i>tms</i>	
------------	--

Returns

int

2.3.2.6 newSimPlaneActor()

```
sim_planeActor * newSimPlaneActor (
    char matriculation[7],
    planeType type,
    unsigned int passengers,
    unsigned int passengersMax,
    planeStatus status )
```

Create a new plane for the simulation.

Parameters

<i>matriculation</i>	
<i>type</i>	
<i>passengers</i>	
<i>passengersMax</i>	
<i>status</i>	

Returns

sim_planeActor*

2.3.2.7 planeNextAction()

```
void planeNextAction (
    airport * airport,
    sim_planeActor * planeActor )
```

Give action to the next plane.

Parameters

<i>airport</i>	
<i>planeActor</i>	

2.4 AirSim.h File Reference

Header for the simulation.

```
#include "AirManager.h"
#include "cmdPrint.h"
```

Data Structures

- struct [simulation](#)
Structure of simulation.
- struct [sim_planeActor](#)
Sstructure of plane Actor.

Functions

- [sim_planeActor * getSimPlaneActorInList](#) ([plane *plane](#), [simulation *simulation](#))
Get the Sim Plane Actor In List object.

2.4.1 Detailed Description

Header for the simulation.

Copyright

Copyright (c) 2021

2.4.2 Function Documentation

2.4.2.1 getSimPlaneActorInList()

```
sim_planeActor* getSimPlaneActorInList (
    plane * plane,
    simulation * simulation )
```

Get the Sim Plane Actor In List object.

Parameters

<i>plane</i>	
<i>simulation</i>	

Returns

sim_planeActor*

2.5 bddManager.c File Reference

continent all functions manipulating the DB: save and load

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "bddManager.h"
```

Functions

- int [CmpPtr](#) (void *data1, void *data2)
compares two function pointers and return 0 or 1
- void [savePlanesInFile](#) (airport *airport)
function that allows to save the airport in a file for later use
- plane * [sMakeChainData](#) (char buffer[100])
retrieves the buffer and puts the info contained in it into a plane which is then returned
- void [openChainFile](#) (char *fileName, simulation *simulation)
function that allows to load the DB
- char * [randomRegistration](#) ()
Allows an automatic generation of matriculation.
- int [randomInt](#) (int min, int max)
a customised randomInt used in randomRegistration

2.5.1 Detailed Description

continent all functions manipulating the DB: save and load

Version

0.1

Date

2021-12-06

Copyright

Copyright (c) 2021

2.5.2 Function Documentation

2.5.2.1 CmpPtr()

```
int CmpPtr (
    void * data1,
    void * data2 )
```

compares two function pointers and return 0 or 1

Parameters

<i>data1</i>	
<i>data2</i>	

Returns

int

2.5.2.2 openChainFile()

```
void openChainFile (
    char * fileName,
    simulation * simulation )
```

function that allows to load the DB

it takes a .txt file and cuts each line, sends it to sMakeChainData which will process it and put each plane in the right place in the right list while putting the right pointers

Parameters

<i>fileName</i>	
<i>simulation</i>	

2.5.2.3 randomInt()

```
int randomInt (
    int min,
    int max )
```

a customised randomInt used in randomRegistration

Parameters

<i>min</i>	
<i>max</i>	

Returns

int

2.5.2.4 randomRegistration()

```
char* randomRegistration ( )
```

Allows an automatic generation of matriculation.

Returns

char*

2.5.2.5 savePlanesInFile()

```
void savePlanesInFile (
    airport * airport )
```

function that allows to save the airport in a file for later use

Parameters

<i>airport</i>	
----------------	--

2.5.2.6 sMakeChainData()

```
plane * sMakeChainData (
    char buffer[100] )
```

retrieves the buffer and puts the info contained in it into a plane which is then returned

Parameters

<i>buffer</i>	
---------------	--

Returns

plane*

2.6 bddManager.h File Reference

groups the function declarations of [bddManager.c](#)

```
#include "AirManager.h"
#include "AirSim.h"
```

Functions

- char * [randomRegistration](#) ()
Allows an automatic generation of matriculation.
- int [randomInt](#) (int min, int max)
a customised randomInt used in randomRegistration
- void [savePlanesInFile](#) ([airport](#) *airport)
function that allows to save the airport in a file for later use
- void [openChainFile](#) (char *fileName, [simulation](#) *simulation)
function that allows to load the DB

2.6.1 Detailed Description

groups the function declarations of [bddManager.c](#)

2.6.2 Function Documentation

2.6.2.1 openChainFile()

```
void openChainFile (
    char * fileName,
    simulation * simulation )
```

function that allows to load the DB

it takes a .txt file and cuts each line, sends it to sMakeChainData which will process it and put each plane in the right place in the right list while putting the right pointers

Parameters

<i>fileName</i>	
<i>simulation</i>	

2.6.2.2 randomInt()

```
int randomInt (
    int min,
    int max )
```

a customised randomInt used in randomRegistration

Parameters

<i>min</i>	
<i>max</i>	

Returns

int

2.6.2.3 randomRegistration()

```
char* randomRegistration ( )
```

Allows an automatic generation of matriculation.

Returns

char*

2.6.2.4 savePlanesInFile()

```
void savePlanesInFile (
    airport * airport )
```

function that allows to save the airport in a file for later use

Parameters

<i>airport</i>	
----------------	--

2.7 cmdPrint.c File Reference

Print the commands on the terminal.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "cmdPrint.h"
```

Functions

- void [debugPlaneType](#) ([planeType](#) plane)
Type of plane.
- void [debugPlaneStatus](#) ([planeStatus](#) plane)
Status of plane.
- void [debugRunwayType](#) ([runwayType](#) runway)
Type of runway.
- void [printPlanesList](#) ([list](#) planesList, bool completeLogs)
Displays the list of aircraft according to the parameters.
- void [printRunwaysList](#) ([list](#) runwaysList, bool completeLogs)
Displays the list of runway according to the parameters.
- void [printParkingsList](#) ([airport](#) airport, bool completeLogs)
Displays the list of the parking according to the parameters.
- void [debugPrintPlane](#) ([plane](#) *plane)
Print informations of each plane in the terminal.
- void [debugPrintRunway](#) ([runway](#) *runway)
Print informations of each runway in the terminal.
- void [debugPrintAirport](#) ([airport](#) airport)
Print informations of the airport in the terminal.

2.7.1 Detailed Description

Print the commands on the terminal.

Copyright

Copyright (c) 2021

2.7.2 Function Documentation

2.7.2.1 debugPlaneStatus()

```
void debugPlaneStatus (  
    planeStatus plane )
```

Status of plane.

Parameters

<i>plane</i>	
--------------	--

2.7.2.2 debugPlaneType()

```
void debugPlaneType (
    planeType plane )
```

Type of plane.

Parameters

<i>plane</i>	
--------------	--

2.7.2.3 debugPrintAirport()

```
void debugPrintAirport (
    airport airport )
```

Print informations of the airport in the terminal.

Parameters

<i>airport</i>	
----------------	--

2.7.2.4 debugPrintPlane()

```
void debugPrintPlane (
    plane * plane )
```

Print informations of each plane in the terminal.

Parameters

<i>plane</i>	
--------------	--

2.7.2.5 debugPrintRunway()

```
void debugPrintRunway (
    runway * runway )
```

Print informations of each runway in the terminal.

Parameters

<i>runway</i>	
---------------	--

2.7.2.6 debugRunwayType()

```
void debugRunwayType (
    runwayType runway )
```

Type of runway.

Parameters

<i>runway</i>	
---------------	--

2.7.2.7 printParkingsList()

```
void printParkingsList (
    airport airport,
    bool completeLogs )
```

Displays the list of the parking according to the parameters.

Print, depending on the parameter (completeLogs), the aircraft registration (false) or all the plane in the parking (true)

If the aircraft registration is yellow, it means that the plane wants to go on the runway queue

Parameters

<i>airport</i>	
<i>completeLogs</i>	

2.7.2.8 printPlanesList()

```
void printPlanesList (
```

```
list planesList,  
bool completeLogs )
```

Displays the list of aircraft according to the parameters.

Print, depending on the parameter (completeLogs), the aircraft registration (false) or all the information about the aircraft (true)

Parameters

<i>planesList</i>	
<i>completeLogs</i>	

2.7.2.9 printRunwaysList()

```
void printRunwaysList (  
    list runwaysList,  
    bool completeLogs )
```

Displays the list of runway according to the parameters.

Print, depending on the parameter (completeLogs), nothing (false) or all the information about the runway (true)

Parameters

<i>runwaysList</i>	
<i>completeLogs</i>	

2.8 cmdPrint.h File Reference

File Header.

```
#include "AirManager.h"
```

Functions

- void `debugPrintPlane` (`plane *plane`)
Print informations of each plane in the terminal.
- void `debugPrintRunway` (`runway *runway`)
Print informations of each runway in the terminal.
- void `debugPrintAirport` (`airport airport`)
Print informations of the airport in the terminal.

2.8.1 Detailed Description

File Header.

Version

0.1

Date

2021-12-05

Copyright

Copyright (c) 2021

2.8.2 Function Documentation

2.8.2.1 debugPrintAirport()

```
void debugPrintAirport (
    airport airport )
```

Print informations of the airport in the terminal.

Parameters

<i>airport</i>	
----------------	--

2.8.2.2 debugPrintPlane()

```
void debugPrintPlane (
    plane * plane )
```

Print informations of each plane in the terminal.

Parameters

<i>plane</i>	
--------------	--

2.8.2.3 debugPrintRunway()

```
void debugPrintRunway (
    runway * runway )
```

Print informations of each runway in the terminal.

Parameters

<i>runway</i>	
---------------	--

2.9 Renderer.c File Reference

Window manager, simulation renderer, temporisation and menus.

```
#include <SDL2/SDL.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>
#include <time.h>
#include "renderer.h"
#include "bddManager.h"
#include <SDL2/SDL_ttf.h>
```

Data Structures

- struct [Anchor](#)
2D Coordinates set
- struct [button](#)
Button item for rendering.

Enumerations

- enum [textAlign](#) { [LEFT](#), [CENTER](#), [RIGHT](#) }
Enumeration of text rendering alignements.
- enum [menuAction](#) {
[MENU_CONTINUE](#), [MENU_SAVE](#), [MENU_OPEN](#), [MENU_QUIT](#),
[MENU_NULL](#) }
Actions of pause menu.

Functions

- void `closeWindow` ()
function that close the window
- void `SetDrawColor` (SDL_Color Color)
Set the Draw Color object.
- void `printRectangleWithBorder` (Anchor CSG, Anchor CID, SDL_Color couleur, int border)
print a rectangle with a customized border into the renderer
- void `printProgress` (Anchor CSG, Anchor CID, SDL_Color couleur, int border, float pourcentage)
print the progress bar into the renderer
- void `printText` (char *text, int fontSize, SDL_Color color, Anchor origin, textAlign align)
print the text you want into the renderer
- void `interf_AirportToRender` (simulation simulation)
print the airport's interface into the renderer
- void `interf_Runway` (simulation simulation, runway *runway, Anchor CSG, int w, int h)
print a generic runway that is used 6times
- void `interf_Radar` (airport *airport, Anchor left)
print the radar square into the renderer
- void `interf_Radar_PrintLine` (plane *plane, Anchor left)
printf the radar intels into the radar square
- void `interf_Parking` (simulation simulation, Anchor left)
print the parking square into the renderer
- void `interf_Parking_PrintLine` (sim_planeActor *planeActor, Anchor left, bool isInWFTR)
printf the parking intels into the radar square
- void `interf_launchMenu` (simulation *simulation)
print the menu into the renderer and do the action when you click on a button
- `button` `newButton` (Anchor center, int h, int w, char *text, menuAction action)
creat a button with a text, a location and a size (width and heigth)
- void `printButtons` (button *buttons, int nButtons)
printf that good looking button in the renderer
- void `updateHoverButtons` (button *buttons, int nButtons, int mx, int my)
add a cool effect (fill the background) when hovering the button
- `menuAction` `getActionButton` (button *buttons, int nButtons)
Get the right Action Button to perform when you click.
- bool `isButtonHover` (int mx, int my, button button)
tells you if your cursor is hovering a button or not
- void `initWindow` (int width, int height)
initialize the window
- void `updateAirportRenderer` (simulation *simulation)
function that refreshes the airport display

Variables

- SDL_Color **YELLOW** = {255, 255, 0, 255}
- SDL_Color **CYAN** = {0, 255, 255, 255}
- SDL_Color **WHITE** = {255, 255, 255, 255}
- SDL_Color **BLACK** = {0, 0, 0, 255}
- SDL_Color **GREY** = {155, 155, 155, 255}
- int **wWidth**
- int **wHeight**
- SDL_Window * **window** = NULL
- SDL_Renderer * **renderer** = NULL

2.9.1 Detailed Description

Window manager, simulation renderer, temporisation and menus.

Copyright

Copyright (c) 2021

2.9.2 Enumeration Type Documentation

2.9.2.1 menuAction

enum `menuAction`

Actions of pause menu.

Enumerator

MENU_CONTINUE	Resume simulation
MENU_SAVE	Save planes status in a file
MENU_OPEN	Open planes status in a file
MENU_QUIT	Quit application
MENU_NULL	No action

2.9.2.2 textAlign

enum `textAlign`

Enumeration of text rendering alignements.

Enumerator

LEFT	Align text to left
CENTER	Align text to center
RIGHT	Align text to right

2.9.3 Function Documentation

2.9.3.1 getActionButton()

```
menuAction getActionButton (
    button * buttons,
    int nButtons )
```

Get the right Action Button to perform when you click.

Parameters

<i>buttons</i>	
<i>nButtons</i>	

Returns

menuAction

2.9.3.2 initWindow()

```
void initWindow (
    int width,
    int height )
```

initialize the window

Parameters

<i>width</i>	
<i>height</i>	

2.9.3.3 interf_AirportToRender()

```
void interf_AirportToRender (
    simulation simulation )
```

print the airport's interface into the renderer

it use 4 subfunction that print each element

Parameters

<i>simulation</i>	
-------------------	--

2.9.3.4 interf_launchMenu()

```
void interf_launchMenu (
    simulation * simulation )
```

print the menu into the renderer and do the action when you click on a button

Parameters

<i>simulation</i>	
-------------------	--

2.9.3.5 interf_Parking()

```
void interf_Parking (
    simulation simulation,
    Anchor left )
```

print the parking square into the renderer

Parameters

<i>simulation</i>	
<i>left</i>	

2.9.3.6 interf_Parking_PrintLine()

```
void interf_Parking_PrintLine (
    sim_planeActor * planeActor,
    Anchor left,
    bool isInWFTR )
```

printf the parking intels into the radar square

Parameters

<i>planeActor</i>	
<i>left</i>	
<i>isInWFTR</i>	

2.9.3.7 interf_Radar()

```
void interf_Radar (
```



```
airport * airport,  
Anchor left )
```

print the radar square into the renderer

Parameters

<i>airport</i>	
<i>left</i>	

2.9.3.8 interf_Radar_PrintLine()

```
void interf_Radar_PrintLine (  
    plane * plane,  
    Anchor left )
```

printf the radar intels into the radar square

Parameters

<i>plane</i>	
<i>left</i>	

2.9.3.9 interf_Runway()

```
void interf_Runway (  
    simulation simulation,  
    runway * runway,  
    Anchor CSG,  
    int w,  
    int h )
```

print a generic runway that is used 6times

Parameters

<i>simulation</i>	
<i>runway</i>	
<i>CSG</i>	
<i>w</i>	
<i>h</i>	

2.9.3.10 isButtonHover()

```
bool isButtonHover (
    int mx,
    int my,
    button button )
```

tells you if your cursor is hovering a button or not

Parameters

<i>mx</i>	
<i>my</i>	
<i>button</i>	

Returns

true
false

2.9.3.11 newButton()

```
button newButton (
    Anchor center,
    int h,
    int w,
    char * text,
    menuAction action )
```

creat a button with a text, a location and a size (width and heigth)

Parameters

<i>center</i>	
<i>h</i>	
<i>w</i>	
<i>text</i>	
<i>action</i>	

Returns

button

2.9.3.12 printButtons()

```
void printButtons (
    button * buttons,
    int nButtons )
```

printf that good looking button in the renderer

Parameters

<i>buttons</i>	
<i>nButtons</i>	

2.9.3.13 printProgress()

```
void printProgress (
    Anchor CSG,
    Anchor CID,
    SDL_Color couleur,
    int border,
    float pourcentage )
```

print the progress bar into the renderer

Parameters

<i>CSG</i>	
<i>CID</i>	
<i>couleur</i>	
<i>border</i>	
<i>pourcentage</i>	

2.9.3.14 printRectangleWithBorder()

```
void printRectangleWithBorder (
    Anchor CSG,
    Anchor CID,
    SDL_Color couleur,
    int border )
```

print a rectangle with a customized border into the renderer

it actually draw a rectangle with 4 little rectangles. this way we can customize the width of the rectangle

Parameters

<i>CSG</i>	
<i>CID</i>	
<i>couleur</i>	
<i>border</i>	

2.9.3.15 printText()

```
void printText (
    char * text,
    int  fontSize,
    SDL_Color color,
    Anchor origin,
    textAlign align )
```

print the text you want into the renderer

Parameters

<i>text</i>	
<i>fontSize</i>	
<i>color</i>	
<i>origin</i>	
<i>align</i>	

2.9.3.16 SetDrawColor()

```
void SetDrawColor (
    SDL_Color Color )
```

Set the Draw Color object.

Parameters

<i>Color</i>	
--------------	--

2.9.3.17 updateAirportRenderer()

```
void updateAirportRenderer (
    simulation * simulation )
```

function that refreshes the airport display

Parameters

<i>simulation</i>	
-------------------	--

2.9.3.18 updateHoverButtons()

```
void updateHoverButtons (
    button * buttons,
    int nButtons,
    int mx,
    int my )
```

add a cool effect (fill the background) when hovering the button

Parameters

<i>buttons</i>	
<i>nButtons</i>	
<i>mx</i>	
<i>my</i>	

2.10 Renderer.h File Reference

Header of [Renderer.c](#).

```
#include "AirSim.h"
```

Functions

- void [initWindow](#) (int width, int height)
initialize the window
- void [updateAirportRenderer](#) ([simulation](#) *simulation)
function that refreshes the airport display

2.10.1 Detailed Description

Header of [Renderer.c](#).

2.10.2 Function Documentation

2.10.2.1 initWindow()

```
void initWindow (
    int width,
    int height )
```

initialize the window

Parameters

<i>width</i>	
<i>height</i>	

2.10.2.2 updateAirportRenderer()

```
void updateAirportRenderer (
    simulation * simulation )
```

function that refreshes the airport display

Parameters

<i>simulation</i>	
-------------------	--

2.11 SmartList.c File Reference

Functions for the management of lists.

```
#include <stdio.h>
#include <stdlib.h>
#include "SmartList.h"
```

Functions

- [chainItem](#) * [newChainItem](#) (void *data)
init chain item
- [list](#) * [newList](#) (compareTwoPointersFunction comparatorFunction)
Allow to create a new list.
- [chainItem](#) * [getItemAtIndex](#) ([list](#) list, int index)
Get the Item At Index object.
- void [updateIndexes](#) ([list](#) *list)
Update indexes of a list.
- void * [getDataAtIndex](#) ([list](#) list, int index)
Get the Data at index in list.
- void [putItemAtIndex](#) ([list](#) *list, [chainItem](#) *item, int index)
Put an item at specified index and update the list architecture to support it.
- void [deleteItemAtIndex](#) ([list](#) *list, int index)
Delete the Item at Index object.
- void [pushInList](#) ([list](#) *list, void *data)
Add the item at the end of the list.
- void [appendInList](#) ([list](#) *list, void *data)
Add the item at the beginning of the list.

- void `appendAtInList` (`list *list`, void *data, int index)
Add the item at the emplacement of the index in the list.
- `chainItem` * `searchItemInList` (`list list`, void *data)
- int `searchIndexInList` (`list list`, void *data)
Search the Index in the list.
- void * `searchDataInList` (`list list`, void *data)
Search the data in the List.
- void `deleteInList` (`list *list`, void *data)
Delete the item in the List.
- void `emptyList` (`list *list`)
Delete All Items in the list.

2.11.1 Detailed Description

Functions for the management of lists.

Copyright

Copyright (c) 2021

2.11.2 Function Documentation

2.11.2.1 `appendAtInList()`

```
void appendAtInList (
    list * list,
    void * data,
    int index )
```

Add the item at the emplacement of the index in the list.

Parameters

<i>list</i>	
<i>data</i>	
<i>index</i>	

2.11.2.2 `appendInList()`

```
void appendInList (
    list * list,
    void * data )
```

Add the item at the beginning of the list.

Parameters

<i>list</i>	
<i>data</i>	

2.11.2.3 deleteInList()

```
void deleteInList (
    list * list,
    void * data )
```

Delete the item in the List.

Parameters

<i>list</i>	
<i>data</i>	

2.11.2.4 deleteItemAtIndex()

```
void deleteItemAtIndex (
    list * list,
    int index )
```

Delete the Item at Index object.

Parameters

<i>list</i>	
<i>index</i>	

2.11.2.5 emptyList()

```
void emptyList (
    list * list )
```

Delete All Items in the list.

Parameters

<i>list</i>	
-------------	--

2.11.2.6 getDataAtIndex()

```
void* getDataAtIndex (
    list list,
    int index )
```

Get the Data at index in list.

Parameters

<i>list</i>	
<i>index</i>	

Returns

pointer to data

2.11.2.7 getItemAtIndex()

```
chainItem* getItemAtIndex (
    list list,
    int index )
```

Get the Item At Index object.

Parameters

<i>list</i>	
<i>index</i>	

Returns

chainItem*

2.11.2.8 newChainItem()

```
chainItem* newChainItem (
    void * data )
```

init chain item

Parameters

<i>in</i>	<i>data</i>	pointer to data to stock in item
-----------	-------------	----------------------------------

Returns

[chainItem](#) pointer to allocated memory for intialized chain item

2.11.2.9 newList()

```
list* newList (
    compareTwoPointersFunction comparatorFunction )
```

Allow to create a new list.

Parameters

<i>comparatorFunction</i>	
---------------------------	--

Returns

list

2.11.2.10 pushInList()

```
void pushInList (
    list * list,
    void * data )
```

Add the item at the end of the list.

Parameters

<i>list</i>	
<i>data</i>	

2.11.2.11 putItemAtIndex()

```
void putItemAtIndex (
    list * list,
```

```
chainItem * item,  
int index )
```

Put an item at specified index and update the list architecture to support it.

Parameters

in	<i>list</i>	
in	<i>item</i>	
in	<i>index</i>	

2.11.2.12 searchDataInList()

```
void* searchDataInList (  
    list list,  
    void * data )
```

Search the data in the List.

Parameters

<i>list</i>	
<i>data</i>	

Returns

void*

2.11.2.13 searchIndexInList()

```
int searchIndexInList (  
    list list,  
    void * data )
```

Search the Index in the list.

Parameters

<i>list</i>	
<i>data</i>	

Returns

int

2.11.2.14 updateIndexs()

```
void updateIndexs (  
    list * list )
```

Update indexes of a list.

Parameters

in	<i>list</i>	list to update
----	-------------	----------------

2.12 SmartList.h File Reference

Manage Lists.

```
#include <stdio.h>
#include <stdlib.h>
```

Data Structures

- struct [chainItem](#)
Item container of list.
- struct [list](#)
List object.

Typedefs

- typedef void(* [voidFunction](#)) (void)
- typedef int(* [compareTwoPointersFunction](#)) (void *data1, void *data2)
- typedef void(* [voidOnePointersFunction](#)) (void *data)
- typedef struct [chainItem](#) [chainItem](#)

Functions

- [list](#) * [newList](#) ([compareTwoPointersFunction](#) comparatorFunction)
Allow to create a new list.
- void * [getDataAtIndex](#) ([list](#) list, int index)
Get the Data at index in list.
- [chainItem](#) * [getItemAtIndex](#) ([list](#) list, int index)
Get the Item At Index object.
- void [deleteItemAtIndex](#) ([list](#) *list, int index)
Delete the Item at Index object.
- void [deleteInList](#) ([list](#) *list, void *data)
Delete the item in the List.
- void [pushInList](#) ([list](#) *list, void *data)
Add the item at the end of the list.
- void [appendInList](#) ([list](#) *list, void *data)
Add the item at the beginning of the list.
- void [appendAtInList](#) ([list](#) *list, void *data, int index)
Add the item at the emplacement of the index in the list.
- int [searchIndexInList](#) ([list](#) list, void *data)
Search the Index in the list.
- void * [searchDataInList](#) ([list](#) list, void *data)
Search the data in the List.
- void [emptyList](#) ([list](#) *list)
Delete All Items in the list.

2.12.1 Detailed Description

Manage Lists.

Copyright

Copyright (c) 2021

2.12.2 Function Documentation

2.12.2.1 appendAtInList()

```
void appendAtInList (
    list * list,
    void * data,
    int index )
```

Add the item at the emplacement of the index in the list.

Parameters

<i>list</i>	
<i>data</i>	
<i>index</i>	

2.12.2.2 appendInList()

```
void appendInList (
    list * list,
    void * data )
```

Add the item at the beginning of the list.

Parameters

<i>list</i>	
<i>data</i>	

2.12.2.3 deleteInList()

```
void deleteInList (
```

```
list * list,  
void * data )
```

Delete the item in the List.

Parameters

<i>list</i>	
<i>data</i>	

2.12.2.4 deleteItemAtIndex()

```
void deleteItemAtIndex (  
    list * list,  
    int index )
```

Delete the Item at Index object.

Parameters

<i>list</i>	
<i>index</i>	

2.12.2.5 emptyList()

```
void emptyList (  
    list * list )
```

Delete All Items in the list.

Parameters

<i>list</i>	
-------------	--

2.12.2.6 getDataAtIndex()

```
void* getDataAtIndex (  
    list list,  
    int index )
```

Get the Data at index in list.

Parameters

<i>list</i>	
<i>index</i>	

Returns

pointer to data

2.12.2.7 getItemAtIndex()

```
chainItem* getItemAtIndex (
    list list,
    int index )
```

Get the Item At Index object.

Parameters

<i>list</i>	
<i>index</i>	

Returns

chainItem*

2.12.2.8 newList()

```
list* newList (
    compareTwoPointersFunction comparatorFunction )
```

Allow to create a new list.

Parameters

<i>comparatorFunction</i>	
---------------------------	--

Returns

list

2.12.2.9 pushInList()

```
void pushInList (
    list * list,
    void * data )
```

Add the item at the end of the list.

Parameters

<i>list</i>	
<i>data</i>	

2.12.2.10 searchDataInList()

```
void* searchDataInList (
    list list,
    void * data )
```

Search the data in the List.

Parameters

<i>list</i>	
<i>data</i>	

Returns

void*

2.12.2.11 searchIndexInList()

```
int searchIndexInList (
    list list,
    void * data )
```

Search the Index in the list.

Parameters

<i>list</i>	
<i>data</i>	

Returns

int