

AirportSimulation

Generated by Doxygen 1.8.11

Contents

| | | |
|----------|--------------------------------------------------|----------|
| 1 | Class Index | 1 |
| 1.1 | Class List | 1 |
| 2 | Class Documentation | 3 |
| 2.1 | Airplane Class Reference | 3 |
| 2.1.1 | Detailed Description | 4 |
| 2.1.2 | Constructor & Destructor Documentation | 4 |
| 2.1.2.1 | Airplane() | 4 |
| 2.1.3 | Member Function Documentation | 4 |
| 2.1.3.1 | approach(std::ostream &log) | 4 |
| 2.1.3.2 | ascend(std::ostream &fLog) | 5 |
| 2.1.3.3 | atRunway(std::ostream &fLog) | 5 |
| 2.1.3.4 | checkFuel(std::ostream &log) | 5 |
| 2.1.3.5 | circle(std::ostream &log) | 5 |
| 2.1.3.6 | complete() const | 6 |
| 2.1.3.7 | crossArrival(std::ostream &log) | 6 |
| 2.1.3.8 | deboard(std::ostream &log) | 6 |
| 2.1.3.9 | decreaseAltitude() | 6 |
| 2.1.3.10 | decreaseTimeRemaining() | 6 |
| 2.1.3.11 | descend(std::ostream &log) | 6 |
| 2.1.3.12 | getFuelCost() | 7 |
| 2.1.3.13 | getSize() const | 7 |
| 2.1.3.14 | increaseAltitude() | 7 |
| 2.1.3.15 | onRunway(std::ostream &fLog) | 7 |

| | | |
|----------|-----------------------------------------------------------------------------------------------------------------------------|----|
| 2.1.3.16 | <code>performNextStep(std::ostream &log)</code> | 7 |
| 2.1.3.17 | <code>prepare(std::ostream &fLog)</code> | 8 |
| 2.1.3.18 | <code>properlyInitialized()</code> <code>const</code> | 8 |
| 2.1.3.19 | <code>pushback(std::ostream &fLog)</code> | 8 |
| 2.1.3.20 | <code>setAltitude(int altitude)</code> | 8 |
| 2.1.3.21 | <code>setCurFuel(int fuel)</code> | 8 |
| 2.1.3.22 | <code>setFuel(int fuel)</code> | 8 |
| 2.1.3.23 | <code>setGateID(int id)</code> | 9 |
| 2.1.3.24 | <code>setPassengers(int fPassengers)</code> | 9 |
| 2.1.3.25 | <code>setSquawk(int squawk)</code> | 9 |
| 2.1.3.26 | <code>setTimeRemaining(int time)</code> | 9 |
| 2.1.3.27 | <code>taxiArrival(std::ostream &log)</code> | 9 |
| 2.1.3.28 | <code>taxiDepartureCross(std::ostream &fLog)</code> | 9 |
| 2.1.3.29 | <code>taxiDepartureStart(std::ostream &fLog)</code> | 10 |
| 2.1.3.30 | <code>taxiDepartureStep(std::ostream &fLog)</code> | 10 |
| 2.1.3.31 | <code>technicalCheck(std::ostream &log)</code> | 10 |
| 2.2 | Airport Class Reference | 10 |
| 2.2.1 | Detailed Description | 11 |
| 2.2.2 | Constructor & Destructor Documentation | 11 |
| 2.2.2.1 | <code>Airport()</code> | 11 |
| 2.2.2.2 | <code>~Airport()</code> | 11 |
| 2.2.3 | Member Function Documentation | 11 |
| 2.2.3.1 | <code>addRunway(Runway *runway)</code> | 11 |
| 2.2.3.2 | <code>amountOfRunways()</code> <code>const</code> | 12 |
| 2.2.3.3 | <code>complete()</code> <code>const</code> | 12 |
| 2.2.3.4 | <code>drawImpression(const Time &time, const std::vector< FlightPlan * > &plans)</code> <code>const</code> | 12 |
| 2.2.3.5 | <code>getFreeGate()</code> | 12 |
| 2.2.3.6 | <code>getFreeRunway(Airplane *plane)</code> <code>const</code> | 12 |
| 2.2.3.7 | <code>getName()</code> <code>const</code> | 13 |
| 2.2.3.8 | <code>getNextRunway(Airplane *airplane)</code> <code>const</code> | 13 |

| | | |
|----------|--------------------------------------------------------------------------------|----|
| 2.2.3.9 | <code>getRunway(const std::string &) const</code> | 13 |
| 2.2.3.10 | <code>graphicsINI(const std::vector< FlightPlan * > &plans)</code> | 13 |
| 2.2.3.11 | <code>initGates()</code> | 14 |
| 2.2.3.12 | <code>properlyInitialized() const</code> | 14 |
| 2.2.3.13 | <code>restoreGate(int id)</code> | 14 |
| 2.2.3.14 | <code>setGates(int gates)</code> | 14 |
| 2.3 | ATC Class Reference | 15 |
| 2.3.1 | Constructor & Destructor Documentation | 15 |
| 2.3.1.1 | <code>ATC(std::ostream &stream)</code> | 15 |
| 2.3.1.2 | <code>~ATC()</code> | 16 |
| 2.3.2 | Member Function Documentation | 16 |
| 2.3.2.1 | <code>doHeartbeat()</code> | 16 |
| 2.3.2.2 | <code>formatMessage(Time time, std::string source, std::string message)</code> | 16 |
| 2.3.2.3 | <code>getNextRequest()</code> | 16 |
| 2.3.2.4 | <code>getQueue()</code> | 16 |
| 2.3.2.5 | <code>getQueueSize() const</code> | 16 |
| 2.3.2.6 | <code>getSquawk(Airplane *airplane)</code> | 17 |
| 2.3.2.7 | <code>getTime() const</code> | 17 |
| 2.3.2.8 | <code>processApproach(Airplane *airplane)</code> | 17 |
| 2.3.2.9 | <code>processDescend(Airplane *airplane)</code> | 17 |
| 2.3.2.10 | <code>processEmergency(Airplane *airplane)</code> | 18 |
| 2.3.2.11 | <code>processIFRClearance(Airplane *airplane)</code> | 18 |
| 2.3.2.12 | <code>processPushback(Airplane *airplane)</code> | 18 |
| 2.3.2.13 | <code>processTakeOff(Airplane *airplane)</code> | 18 |
| 2.3.2.14 | <code>processTakeOffRunway(Airplane *airplane)</code> | 18 |
| 2.3.2.15 | <code>processTaxiArrival(Airplane *airplane)</code> | 19 |
| 2.3.2.16 | <code>processTaxiInitialise(Airplane *airplane)</code> | 19 |
| 2.3.2.17 | <code>processTaxiInstruction(Airplane *airplane)</code> | 19 |
| 2.3.2.18 | <code>processUrgentEmergency(Airplane *airplane)</code> | 19 |
| 2.3.2.19 | <code>properlyInitialized() const</code> | 20 |

| | | |
|----------|----------------------------------------------------------|----|
| 2.3.2.20 | <code>sendMessage(const std::string &message)</code> | 20 |
| 2.3.2.21 | <code>sendRequest(Time time, Airplane *source)</code> | 20 |
| 2.3.2.22 | <code>setTime(Time time)</code> | 20 |
| 2.4 | ATCRequest Struct Reference | 21 |
| 2.4.1 | Constructor & Destructor Documentation | 21 |
| 2.4.1.1 | <code>ATCRequest(Time time, Airplane *plane)</code> | 21 |
| 2.4.2 | Member Data Documentation | 21 |
| 2.4.2.1 | <code>fAirplane</code> | 21 |
| 2.4.2.2 | <code>fTime</code> | 21 |
| 2.5 | Comparator Struct Reference | 22 |
| 2.6 | FlightPlan Class Reference | 22 |
| 2.6.1 | Detailed Description | 22 |
| 2.6.2 | Constructor & Destructor Documentation | 22 |
| 2.6.2.1 | <code>FlightPlan()</code> | 22 |
| 2.6.2.2 | <code>~FlightPlan()</code> | 23 |
| 2.6.3 | Member Function Documentation | 23 |
| 2.6.3.1 | <code>complete() const</code> | 23 |
| 2.6.3.2 | <code>getDestination() const</code> | 23 |
| 2.6.3.3 | <code>getEvent(Time time)</code> | 23 |
| 2.6.3.4 | <code>properlyInitialized() const</code> | 23 |
| 2.6.3.5 | <code>setArrival(int arrival)</code> | 23 |
| 2.6.3.6 | <code>setDeparture(int departure)</code> | 24 |
| 2.6.3.7 | <code>setInterval(int interval)</code> | 24 |
| 2.7 | Graphics Class Reference | 24 |
| 2.7.1 | Detailed Description | 24 |
| 2.7.2 | Constructor & Destructor Documentation | 25 |
| 2.7.2.1 | <code>Graphics(Airport *airport)</code> | 25 |
| 2.7.3 | Member Function Documentation | 25 |
| 2.7.3.1 | <code>addElement(Airplane *airplane)</code> | 25 |
| 2.7.3.2 | <code>addElement(Runway *runway)</code> | 25 |

| | | |
|----------|------------------------------------------------------------------------------------------|----|
| 2.7.3.3 | <code>generateINI(double x, double y, double z, int size=3000) const</code> | 25 |
| 2.7.3.4 | <code>properlyInitialized() const</code> | 26 |
| 2.8 | Input Class Reference | 26 |
| 2.8.1 | Detailed Description | 26 |
| 2.8.2 | Constructor & Destructor Documentation | 27 |
| 2.8.2.1 | <code>Input()</code> | 27 |
| 2.8.3 | Member Function Documentation | 27 |
| 2.8.3.1 | <code>addAirport(Airport *airport)</code> | 27 |
| 2.8.3.2 | <code>addFlightPlan(FlightPlan *flightPlan)</code> | 27 |
| 2.8.3.3 | <code>addRunway(Runway *runway)</code> | 27 |
| 2.8.3.4 | <code>findAirportByIATA(const std::string &iata) const</code> | 27 |
| 2.8.3.5 | <code>getAirports() const</code> | 28 |
| 2.8.3.6 | <code>getFlightPlans() const</code> | 28 |
| 2.8.3.7 | <code>isNumber(const std::string &input)</code> | 28 |
| 2.8.3.8 | <code>properlyInitialized() const</code> | 28 |
| 2.8.3.9 | <code>read(const std::string &filename, std::ostream &errorLog=std::cerr)</code> | 29 |
| 2.9 | Runway Class Reference | 29 |
| 2.9.1 | Detailed Description | 29 |
| 2.9.2 | Constructor & Destructor Documentation | 29 |
| 2.9.2.1 | <code>Runway()</code> | 29 |
| 2.9.3 | Member Function Documentation | 30 |
| 2.9.3.1 | <code>complete() const</code> | 30 |
| 2.9.3.2 | <code>getType() const</code> | 30 |
| 2.9.3.3 | <code>properlyInitialized() const</code> | 30 |
| 2.9.3.4 | <code>validForAirplane(Airplane *plane) const</code> | 30 |
| 2.10 | RunwayInfo Struct Reference | 31 |
| 2.10.1 | Detailed Description | 31 |
| 2.10.2 | Member Data Documentation | 31 |
| 2.10.2.1 | <code>arrivingPlanes</code> | 31 |
| 2.10.2.2 | <code>departingPlanes</code> | 31 |

| | | |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.10.2.3 | runway | 31 |
| 2.11 | System Class Reference | 32 |
| 2.11.1 | Detailed Description | 32 |
| 2.11.2 | Constructor & Destructor Documentation | 32 |
| 2.11.2.1 | System(const Input &input, std::ostream &atc, const Time &end) | 32 |
| 2.11.2.2 | ~System() | 32 |
| 2.11.3 | Member Function Documentation | 32 |
| 2.11.3.1 | generateImages(Time start, Time end) | 33 |
| 2.11.3.2 | getAirport() const | 33 |
| 2.11.3.3 | getATC() const | 33 |
| 2.11.3.4 | getFlightPlans() const | 33 |
| 2.11.3.5 | info(std::ostream &out) | 33 |
| 2.11.3.6 | properlyInitialized() const | 34 |
| 2.11.3.7 | run(std::ostream &log, const std::string &impressionName="../../output/impressions/impression", const std::string &iniName="../../output/ini/graphics") | 34 |
| 2.11.3.8 | simulationFinished() const | 34 |
| 2.12 | Time Class Reference | 34 |
| 2.12.1 | Detailed Description | 35 |
| 2.12.2 | Constructor & Destructor Documentation | 35 |
| 2.12.2.1 | Time(int hour=12, int minute=0) | 35 |
| 2.12.2.2 | Time(const Time &time) | 35 |
| 2.12.3 | Member Function Documentation | 35 |
| 2.12.3.1 | advance(int minutes=1) | 36 |
| 2.12.3.2 | formatted() const | 36 |
| 2.12.3.3 | getHour() const | 36 |
| 2.12.3.4 | getMinute() const | 36 |
| 2.12.3.5 | operator<(const Time &) const | 36 |
| 2.12.3.6 | operator=(const Time &time) | 36 |
| 2.12.3.7 | operator==(const Time &) const | 37 |
| 2.12.3.8 | properlyInitialized() const | 37 |
| 2.12.3.9 | setHour(int hour) | 37 |
| 2.12.3.10 | setMinute(int minute) | 37 |

Chapter 1

Class Index

1.1 Class List

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

| | | |
|----------------------------|-------------------------------------------------------------------|----|
| Airplane | : Class that represents an airplane in the simulation | 3 |
| Airport | | 10 |
| ATC | | 15 |
| ATCRequest | | 21 |
| Comparator | | 22 |
| FlightPlan | | |
| | : Class that represents a flight plan in the simulation | 22 |
| Graphics | | 24 |
| Input | | |
| | : Class that reads input for the simulation | 26 |
| Runway | | 29 |
| RunwayInfo | | 31 |
| System | | |
| | : Main class, controls the simulation | 32 |
| Time | | 34 |

Chapter 2

Class Documentation

2.1 Airplane Class Reference

: Class that represents an airplane in the simulation

```
#include <Airplane.h>
```

Public Member Functions

- [Airplane](#) ()
- bool [properlyInitialized](#) () const
- void [performNextStep](#) (std::ostream &log)
- bool [complete](#) () const
- int [getFuelCost](#) ()
- void [increaseAltitude](#) ()
- void [decreaseAltitude](#) ()
- void [decreaseTimeRemaining](#) ()
- void [approach](#) (std::ostream &log)
- void [descend](#) (std::ostream &log)
- void [circle](#) (std::ostream &log)
- void [taxiArrival](#) (std::ostream &log)
- void [crossArrival](#) (std::ostream &log)
- void [deboard](#) (std::ostream &log)
- void [technicalCheck](#) (std::ostream &log)
- void [ascend](#) (std::ostream &fLog)
- void [onRunway](#) (std::ostream &fLog)
- void [atRunway](#) (std::ostream &fLog)
- void [taxiDepartureCross](#) (std::ostream &fLog)
- void [taxiDepartureStep](#) (std::ostream &fLog)
- void [taxiDepartureStart](#) (std::ostream &fLog)
- void [pushback](#) (std::ostream &fLog)
- void [prepare](#) (std::ostream &fLog)
- void [checkFuel](#) (std::ostream &log)
- void [setAltitude](#) (int altitude)
Getters and setters with special contracts/documentation.
- void [setTimeRemaining](#) (int time)
- void [setPassengers](#) (int fPassengers)

- void [setGateID](#) (int id)
- void [setFuel](#) (int fuel)
- void [setSquawk](#) (int squawk)
- void [setCurFuel](#) (int fuel)
- EPlaneSize [getSize](#) () const
- void **setSize** (EPlaneSize size)
- EPlaneType **getType** () const
- void **setType** (EPlaneType type)
- EPlaneEngine **getEngine** () const
- void **setEngine** (EPlaneEngine engine)
- int **getAltitude** () const
- int **getGateID** () const
- int **getPassengers** () const
- const std::string & **getNumber** () const
- void **setNumber** (const std::string &fNumber)
- const std::string & **getCallsign** () const
- void **setCallsign** (const std::string &fCallsign)
- const std::string & **getModel** () const
- void **setModel** (const std::string &fModel)
- EPlaneStatus **getStatus** () const
- void **setStatus** (EPlaneStatus fStatus)
- int **getTimeRemaining** () const
- void **setPosition** (const std::string &)
- const std::string & **getPosition** () const
- void **setRequest** (EPlaneRequest)
- EPlaneRequest **getRequest** () const
- [Runway](#) * **getRunway** () const
- void **setRunway** ([Runway](#) *)
- int **getFuel** () const
- int **getCurFuel** () const
- int **getSquawk** () const
- void **setATC** ([ATC](#) *atc)

2.1.1 Detailed Description

: Class that represents an airplane in the simulation

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Airplane::Airplane ()

Default constructor

ENSURE([properlyInitialized\(\)](#), "constructor must end in properlyInitialized state");

2.1.3 Member Function Documentation

2.1.3.1 void Airplane::approach (std::ostream & log)

Flying functions Only use when really needed, else use [Airplane::performNextStep](#) Performs the approach of a given plane.

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane wasn't initialized when calling approach");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.2 void Airplane::ascend (std::ostream & *fLog*)

Ascends an airplane to 5000ft.

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane was not properly initialized when calling ascend.");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.3 void Airplane::atRunway (std::ostream & *fLog*)

Lets an airplane wait AT a runway until instructions are given.

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane was not properly initialized when calling atRunway");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.4 void Airplane::checkFuel (std::ostream & *log*)

Check fuel, subtract fuelcost if needed, adjust squawk if needed, contact [ATC](#) if needed.

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane was not properly initialized when calling checkFuel");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.5 void Airplane::circle (std::ostream & *log*)

Performs the circling of a given plane.

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane wasn't initialized when calling circle");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.6 bool Airplane::complete () const

Checks if all the data members were initialized

REQUIRE(this->properlyInitialized(), "Airplane was't initialized when calling Airplane::complete");

Returns

boolean isComplete

2.1.3.7 void Airplane::crossArrival (std::ostream & log)

Crosses a runway

REQUIRE(this->properlyInitialized(), "Airplane was't initialized when calling crossArrival");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.8 void Airplane::deboard (std::ostream & log)

Performs the deboarding of a given plane.

REQUIRE(this->properlyInitialized(), "Airplane was't initialized when calling deboard");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.9 void Airplane::decreaseAltitude ()

Decreases the plane's altitude by 1000

REQUIRE(this->properlyInitialized(), "Airplane was't initialized when calling decreaseAltitude");

2.1.3.10 void Airplane::decreaseTimeRemaining ()

Decreases the time of the remaining operation by one.

If there's no operation busy, it does nothing.

REQUIRE(this->properlyInitialized(), "Airplane was't initialized when calling decreaseTimeRemaining");

2.1.3.11 void Airplane::descend (std::ostream & log)

Performs the descend of a given plane.

REQUIRE(this->properlyInitialized(), "Airplane was't initialized when calling descend");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.12 int Airplane::getFuelCost ()

Calculate the fuel cost for a plane

```
REQUIRE(this->properlyInitialized(), "Airplane was not properly initialized when calling getFuelCost");  
REQUIRE(getEngine != kDefaultEngine, "Invalid engine type for calculating fuel.");  
REQUIRE(getSize != kDefaultSize, "Invalid size for calculating fuel.");
```

Returns

fuel cost

2.1.3.13 EPlaneSize Airplane::getSize () const

Getters and setters For each method: REQUIRE(properlyInitialized(), "Airplane wasn't initialized when calling Airplane getter/setter");

2.1.3.14 void Airplane::increaseAltitude ()

Increases the plane's altitude by 1000

```
REQUIRE(this->properlyInitialized(), "Airplane wasn't initialized when calling increaseAltitude");
```

2.1.3.15 void Airplane::onRunway (std::ostream & fLog)

Lets an airplane wait ON a runway before taking off.

```
REQUIRE(this->properlyInitialized(), "Airplane was not properly initialized when calling onRunway.");
```

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.16 void Airplane::performNextStep (std::ostream & log)

Calls the right flying function according to the status.

At the end, checks fuel and decreases time remaining

```
REQUIRE(properlyInitialized(), "Plane was not properly initialized when calling performNextStep.");
```

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.17 void Airplane::prepare (std::ostream & *fLog*)

Refuels the airplane and lets passengers board.

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane was not properly initialized when calling prepare");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.18 bool Airplane::properlyInitialized () const

Checks if the object is properly initialized

2.1.3.19 void Airplane::pushback (std::ostream & *fLog*)

Pushes an airplane back from the gate.

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane was not properly initialized when calling pushback");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.20 void Airplane::setAltitude (int *altitude*)

Getters and setters with special contracts/documentation.

Setter for the altitude

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane wasn't initialized when calling Airplane getter/setter");

REQUIRE(altitude >= 0, "Altitude can't be negative");

2.1.3.21 void Airplane::setCurFuel (int *fuel*)

Setter for the current amount of fuel

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane wasn't initialized when calling Airplane getter/setter");

REQUIRE(fuel >= 0, "Fuel can't be negative");

REQUIRE(fuel <= getFuel, "Fuel can't be more than the max. fuel");

2.1.3.22 void Airplane::setFuel (int *fuel*)

Setter for the maximum amount of fuel (in 10.000 units)

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane wasn't initialized when calling Airplane getter/setter");

REQUIRE(fuel > 0, "Fuel can't be less than 1");

2.1.3.23 void Airplane::setGateID (int *id*)

Setter for the gate the plane's at

```
REQUIRE(this->properlyInitialized(), "Airplane wasn't initialized when calling Airplane getter/setter");  
REQUIRE(id >= -1, "Gate id can't be less than -1");
```

2.1.3.24 void Airplane::setPassengers (int *fPassengers*)

Setter for the maximum amount of passengers

```
REQUIRE(this->properlyInitialized(), "Airplane wasn't initialized when calling Airplane getter/setter");  
REQUIRE(passengers >= 0, "Passenger amount can't be negative");
```

2.1.3.25 void Airplane::setSquawk (int *squawk*)

Setter for the squawk code

```
REQUIRE(this->properlyInitialized(), "Airplane wasn't initialized when calling Airplane getter/setter");  
REQUIRE(squawk >= 0, "Squawk code can't be negative");
```

2.1.3.26 void Airplane::setTimeRemaining (int *time*)

Setter for the time remaining on the operation

```
REQUIRE(this->properlyInitialized(), "Airplane wasn't initialized when calling Airplane getter/setter");  
REQUIRE(time >= 0, "Time remaining can't be negative");
```

2.1.3.27 void Airplane::taxiArrival (std::ostream & *log*)

Performs the taxiing upon arrival of a given plane.

```
REQUIRE(this->properlyInitialized(), "Airplane wasn't initialized when calling taxiArrival");
```

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.28 void Airplane::taxiDepartureCross (std::ostream & *fLog*)

Performs a cross.

```
REQUIRE(this->properlyInitialized(), "Airplane was not properly initialized when calling taxiDepartureCross");
```

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.29 void Airplane::taxiDepartureStart (std::ostream & *fLog*)

Lets an airplane wait before taxiing.

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane was not properly initialized when calling taxiDepartureStart");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.30 void Airplane::taxiDepartureStep (std::ostream & *fLog*)

Performs the next taxi-step.

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane was not properly initialized when calling taxiDepartureStep");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

2.1.3.31 void Airplane::technicalCheck (std::ostream & *log*)

Performs the technical check of the plane

REQUIRE(this->[properlyInitialized\(\)](#), "Airplane wasn't initialized when calling technicalCheck");

Parameters

| | |
|------------|---------------------------------|
| <i>log</i> | ostream used for logging events |
|------------|---------------------------------|

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

- /home/max/CLionProjects/ProjectVliegveld/headers/Airplane.h
- /home/max/CLionProjects/ProjectVliegveld/src/Airplane.cpp

2.2 Airport Class Reference

```
#include <Airport.h>
```

Public Member Functions

- [Airport](#) ()
- [~Airport](#) ()
- bool [properlyInitialized](#) () const
- bool [complete](#) () const
- void [initGates](#) ()
- int [getFreeGate](#) ()

- void `restoreGate` (int id)
- `Runway * getRunway` (const std::string &) const
- `Runway * getNextRunway` (`Airplane *airplane`) const
- void `addRunway` (`Runway *runway`)
- `Runway * getFreeRunway` (`Airplane *plane`) const
- size_t `amountOfRunways` () const
- std::string `drawImpression` (const `Time` &time, const std::vector< `FlightPlan` * > &plans) const
- std::string `graphicsINI` (const std::vector< `FlightPlan` * > &plans)
- void `setGates` (int gates)
- const std::string & `getName` () const
- void `setName` (const std::string &fName)
- const std::string & `getIata` () const
- void `setIata` (const std::string &fIata)
- const std::string & `getCallsign` () const
- void `setCallsign` (const std::string &fCallsign)
- int `getGates` () const
- std::vector< `Runway` * > `getRunways` () const
- std::map< int, bool > `getGateMap` () const

2.2.1 Detailed Description

Class that represents the airport in a simulation

2.2.2 Constructor & Destructor Documentation

2.2.2.1 `Airport::Airport ()`

Default constructor

ENSURE(`properlyInitialized()`, "Airport wasn't properly initialized after constructing");

2.2.2.2 `Airport::~~Airport ()`

Destructor

2.2.3 Member Function Documentation

2.2.3.1 `void Airport::addRunway (Runway * runway)`

Adds a runway to the airport. Present is a boolean indicating if the object is already in the `getRunways()`
 REQUIRE(`properlyInitialized()`, "Airport wasn't properly initialized when calling addRunway.");
 REQUIRE(`runway != NULL`, "Runway cannot be NULL.");
 ENSURE(!`present`, "Runway is already in system.");
 ENSURE(`getRunways().back() == runway`, "Runway was not properly added to the system.");

Parameters

| | |
|---------------------|-------------------------------------------|
| <code>runway</code> | Pointer to runway that needs to be added. |
|---------------------|-------------------------------------------|

2.2.3.2 `size_t Airport::amountOfRunways () const`

Return the amount of runways the airport has.

REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling amountOfRunways.");

Returns

: Amount of runways.

2.2.3.3 `bool Airport::complete () const`

Checks if all the data members were initialized

REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling complete.");

Returns

: Boolean indicating if everything was initialized.

2.2.3.4 `string Airport::drawImpression (const Time & time, const std::vector< FlightPlan * > & plans) const`

Generates a graphical impression for the current state of the airport.

REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling drawImpression.");

Parameters

| | |
|--------------|---------------------------------------|
| <i>time</i> | time of the impression. |
| <i>plans</i> | the flight plans to access the planes |

Returns

string containing the impression.

2.2.3.5 `int Airport::getFreeGate ()`

Returns the ID of a free gate

REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling getFreeGate.");

REQUIRE(getGates() > 0, "Airport has no gates.");

REQUIRE(!getGateMap().empty(), "Gate map not initialized yet");

Returns

: ID of a free gate. -1 if nothing available

2.2.3.6 `Runway * Airport::getFreeRunway (Airplane * plane) const`

Looks for a runway that is free and is suitable for a given airplane.

REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling getFreeRunway.");

REQUIRE(plane != NULL, "Plane object does not exist.");

Parameters

| | |
|--------------|------------------------------|
| <i>plane</i> | Plane that needs the runway. |
|--------------|------------------------------|

Returns

: Pointer to runway, NULL if nothing is found.

2.2.3.7 `const string & Airport::getName () const`

Getters and setters For all: REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling getter/setter."); For setters: ENSURE(getField() == value, "Field wasn't set properly"); Where getField() is flexible

2.2.3.8 `Runway * Airport::getNextRunway (Airplane * airplane) const`

Gets the next runway (for taxiing) based on what the airplane is doing.

REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling getNextRunway.");

REQUIRE([airplane != NULL](#), "Airplane cannot be NULL.");

Parameters

| | |
|-----------------|-------------------------------------------|
| <i>airplane</i> | Airplane that is taxiing. |
|-----------------|-------------------------------------------|

Returns

: Pointer to the found runway, NULL if nothing is found.

2.2.3.9 `Runway * Airport::getRunway (const std::string &) const`

Searches the runway with the given taxipoint.

REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling getRunway.");

Returns

: Pointer to the found runway, NULL if nothing is found.

2.2.3.10 `std::string Airport::graphicsINI (const std::vector< FlightPlan * > & plans)`

Generates a string containing the info for use with the graphics engine

REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling graphicsINI.");

Parameters

| | |
|--------------|------------------|
| <i>plans</i> | the flight plans |
|--------------|------------------|

Returns

string containing ini file

2.2.3.11 void Airport::initGates ()

Initializes the gateStack

REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling initGates.");
REQUIRE(getGateMap().empty(), "Can't initialize gate map, already in use.");

2.2.3.12 bool Airport::properlyInitialized () const

Checks if the object is properly initialized

Returns

: Boolean indicating if properly initialized or not.

2.2.3.13 void Airport::restoreGate (int *id*)

Restores a gate, making it available for use again.

REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling restoreGate.");
REQUIRE(id <= getGates() && id > 0, "Gate ID is invalid.");
REQUIRE(getGateMap()[id], "Gate has to be in use to restore it");

Parameters

| | |
|-----------|-----------------------|
| <i>id</i> | ID of gate to restore |
|-----------|-----------------------|

2.2.3.14 void Airport::setGates (int *gates*)

Set the amount of gates in the airport.

REQUIRE([properlyInitialized\(\)](#), "Airport wasn't properly initialized when calling getter/setter.");
REQUIRE(getGates() >= 0, "Number of gates cannot be negative!");
ENSURE(getGates() == gates, "Field wasn't set properly");

Parameters

| | |
|--------------|------------------|
| <i>gates</i> | Amount of gates. |
|--------------|------------------|

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

- /home/max/CLionProjects/ProjectVliegveld/headers/Airport.h
- /home/max/CLionProjects/ProjectVliegveld/src/Airport.cpp

2.3 ATC Class Reference

Public Member Functions

- [ATC](#) (std::ostream &stream)
- [~ATC](#) ()
- void [doHeartbeat](#) ()
- bool [properlyInitialized](#) () const
- void [sendRequest](#) (Time time, Airplane *source)
- int [getQueueSize](#) () const
- ATCRequest * [getNextRequest](#) ()
- void [processApproach](#) (Airplane *airplane)
- void [processDescend](#) (Airplane *airplane)
- void [processTaxiArrival](#) (Airplane *airplane)
- void [processIFRClearance](#) (Airplane *airplane)
- void [processPushback](#) (Airplane *airplane)
- void [processTaxiInitialise](#) (Airplane *airplane)
- void [processTaxiInstruction](#) (Airplane *airplane)
- void [processTakeOff](#) (Airplane *airplane)
- void [processTakeOffRunway](#) (Airplane *airplane)
- void [processEmergency](#) (Airplane *airplane)
- void [processUrgentEmergency](#) (Airplane *airplane)
- void [sendMessage](#) (const std::string &message)
- int [getSquawk](#) (Airplane *airplane)
- Time [getTime](#) () const
- void [setTime](#) (Time time)
- std::priority_queue< ATCRequest *, std::vector< ATCRequest * >, Comparator > * [getQueue](#) ()
- Airport * [getAirport](#) () const
- void [setAirport](#) (Airport *)
- bool [get3occupied](#) () const
- void [set3occupied](#) (bool)
- bool [get5occupied](#) () const
- void [set5occupied](#) (bool)
- void [setTestMode](#) (bool)
- bool [isTestMode](#) () const

Static Public Member Functions

- static std::string [formatMessage](#) (Time time, std::string source, std::string message)

2.3.1 Constructor & Destructor Documentation

2.3.1.1 ATC::ATC (std::ostream & *stream*)

Constructor

ENSURE([properlyInitialized\(\)](#), "ATC was not properly initialized after constructing.");

Parameters

| | |
|---------------|---------------------------|
| <i>stream</i> | std::ostream to write to. |
|---------------|---------------------------|

2.3.1.2 ATC::~~ATC ()

Destructor

2.3.2 Member Function Documentation

2.3.2.1 void ATC::doHeartbeat ()

Main function of the [ATC](#), needs to be called every time we advance in time.

Handles requests and responds correctly.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling doHeartbeat.");

2.3.2.2 string ATC::formatMessage (Time *time*, std::string *source*, std::string *message*) [static]

Creates a correctly formatted [ATC](#) message when the contents are given.

Parameters

| | |
|----------------|----------------------------------|
| <i>time</i> | Time of message. |
| <i>source</i> | Sender of message. |
| <i>message</i> | Content of message. |

Returns

: Formatted message.

2.3.2.3 ATCRequest * ATC::getNextRequest ()

Get the next request that needs to be handled by the [ATC](#).

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling getNextRequest.");

Returns

: Pointer to the request, NULL if `priority_queue` is empty.

2.3.2.4 priority_queue< ATCRequest *, vector< ATCRequest * >, Comparator > * ATC::getQueue ()

Getters and setters for the fields of the class. For all: REQUIRE([properlyInitialized\(\)](#), "ATC wasn't properly initialized when calling getter/setter."); For setters; ENSURE(getField == value, "Field wasn't std::set properly"); where get↔Field is specific for the member

2.3.2.5 int ATC::getQueueSize () const

Return the amount of requests that are queued.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling getQueueSize.");

Returns

: Size of the `std::priority_queue`.

2.3.2.6 int ATC::getSquawk (Airplane * *airplane*)

Generates a squawk code for a given plane. The returned code will not be generated for any other plane.
 REQUIRE(this->properlyInitialized(), "ATC was not properly initialized when calling getSquawk.");

Parameters

| | |
|-----------------|----------------------------------|
| <i>airplane</i> | airplane to generate squawk for. |
|-----------------|----------------------------------|

Returns

squawk code

2.3.2.7 Time ATC::getTime () const

Getter for the current time
 REQUIRE(this->properlyInitialized(), "ATC wasn't initialized when calling getTime");

Returns

current time

2.3.2.8 void ATC::processApproach (Airplane * *airplane*)

Processes a request for approach.
 REQUIRE(this->properlyInitialized(), "ATC was not properly initialized when calling processApproach.");

Parameters

| | |
|-----------------|---------------------------|
| <i>airplane</i> | the sender of the request |
| <i>time</i> | current time |

2.3.2.9 void ATC::processDescend (Airplane * *airplane*)

Processes a request for descend.
 REQUIRE(this->properlyInitialized(), "ATC was not properly initialized when calling processDescend.");

Parameters

| | |
|-----------------|---------------------------|
| <i>airplane</i> | the sender of the request |
| <i>time</i> | current time |

2.3.2.10 void ATC::processEmergency (Airplane * airplane)

Processes a request for an emergency landing.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling processTakeoffRunway.");

Parameters

| | |
|-----------------|---------------------------|
| <i>airplane</i> | the sender of the request |
| <i>time</i> | current time |

2.3.2.11 void ATC::processIFRClearance (Airplane * airplane)

Processes a request for IFR clearance.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling processIFRClearancy.");

Parameters

| | |
|-----------------|---------------------------|
| <i>airplane</i> | the sender of the request |
| <i>time</i> | current time |

2.3.2.12 void ATC::processPushback (Airplane * airplane)

Processes a request for pushback.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling processPushback.");

Parameters

| | |
|-----------------|---------------------------|
| <i>airplane</i> | the sender of the request |
| <i>time</i> | current time |

2.3.2.13 void ATC::processTakeOff (Airplane * airplane)

Processes a request for takeoff when waiting at runway.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling processTakeoff.");

Parameters

| | |
|-----------------|---------------------------|
| <i>airplane</i> | the sender of the request |
| <i>time</i> | current time |

2.3.2.14 void ATC::processTakeOffRunway (Airplane * airplane)

Processes a request for takeoff when waiting on runway.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling processTakeoffRunway.");

Parameters

| | |
|-----------------|---------------------------|
| <i>airplane</i> | the sender of the request |
| <i>time</i> | current time |

2.3.2.15 void ATC::processTaxiArrival (Airplane * airplane)

Processes a request for taxiing at arrival.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling processTaxiArrival.");

Parameters

| | |
|-----------------|---------------------------|
| <i>airplane</i> | the sender of the request |
| <i>time</i> | current time |

2.3.2.16 void ATC::processTaxiInitialise (Airplane * airplane)

Processes a request to start taxiing.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling processTaxiInitialise.");

Parameters

| | |
|-----------------|---------------------------|
| <i>airplane</i> | the sender of the request |
| <i>time</i> | current time |

2.3.2.17 void ATC::processTaxiInstruction (Airplane * airplane)

Processes a request for a taxi instruction.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling processTaxiInstruction.");

Parameters

| | |
|-----------------|---------------------------|
| <i>airplane</i> | the sender of the request |
| <i>time</i> | current time |

2.3.2.18 void ATC::processUrgentEmergency (Airplane * airplane)

Processes a request for an urgent emergency landing.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling processUrgentEmergency.");

Parameters

| | |
|-----------------|---------------------------|
| <i>airplane</i> | the sender of the request |
| <i>time</i> | current time |

2.3.2.19 bool ATC::properlyInitialized () const

Checks if the object is properly initialized

Returns

: Boolean indicating if properly initialized or not.

2.3.2.20 void ATC::sendMessage (const std::string & message)

Write a message to the [ATC](#) stream.

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling sendMessage.");

Parameters

| | |
|----------------|--------------------------------|
| <i>message</i> | Message that needs to be send. |
|----------------|--------------------------------|

2.3.2.21 void ATC::sendRequest (Time time, Airplane * source)

Send a request to the [ATC](#).

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was not properly initialized when calling sendRequest.");

REQUIRE(source != NULL, "Source is NULL.");

Parameters

| | |
|---------------|-------------------------------------------------|
| <i>time</i> | Time of the request. |
| <i>source</i> | Airplane that made the request. |

2.3.2.22 void ATC::setTime (Time time)

Setter for the current time

REQUIRE(this->[properlyInitialized\(\)](#), "ATC was't initialized when calling setTime");

Parameters

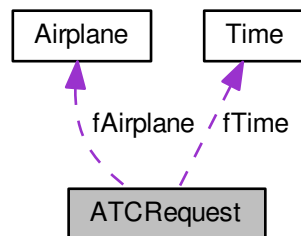
| | |
|-------------|-------------|
| <i>time</i> | time to set |
|-------------|-------------|

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

- /home/max/CLionProjects/ProjectVliegveld/headers/ATC.h
- /home/max/CLionProjects/ProjectVliegveld/src/ATC.cpp

2.4 ATCRequest Struct Reference

Collaboration diagram for ATCRequest:



Public Member Functions

- [ATCRequest](#) ([Time](#) time, [Airplane](#) *plane)

Public Attributes

- [Time](#) fTime
- [Airplane](#) * fAirplane

2.4.1 Constructor & Destructor Documentation

2.4.1.1 `ATCRequest::ATCRequest (Time time, Airplane * plane)`

Constructor.

2.4.2 Member Data Documentation

2.4.2.1 `Airplane* ATCRequest::fAirplane`

[Airplane](#) that sent request

2.4.2.2 `Time ATCRequest::fTime`

[Time](#) message was sent.

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

- `/home/max/CLionProjects/ProjectVliegveld/headers/ATC.h`
- `/home/max/CLionProjects/ProjectVliegveld/src/ATC.cpp`

2.5 Comparator Struct Reference

Public Member Functions

- bool **operator()** (const [ATCRequest](#) *lhs, const [ATCRequest](#) *rhs)

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

- /home/max/CLionProjects/ProjectVliegveld/headers/ATC.h
- /home/max/CLionProjects/ProjectVliegveld/src/ATC.cpp

2.6 FlightPlan Class Reference

: Class that represents a flight plan in the simulation

```
#include <FlightPlan.h>
```

Public Member Functions

- [FlightPlan](#) ()
- [~FlightPlan](#) ()
- bool [properlyInitialized](#) () const
- void [setDeparture](#) (int departure)
- void [setArrival](#) (int arrival)
- void [setInterval](#) (int interval)
- EEvent [getEvent](#) ([Time](#) time)
- bool [complete](#) () const
- const std::string & [getDestination](#) () const
- void **setDestination** (const std::string &fDestination)
- int **getDeparture** () const
- int **getArrival** () const
- int **getInterval** () const
- void **setAirplane** ([Airplane](#) *)
- [Airplane](#) * **getAirplane** () const

2.6.1 Detailed Description

: Class that represents a flight plan in the simulation

2.6.2 Constructor & Destructor Documentation

2.6.2.1 FlightPlan::FlightPlan ()

Default constructor

```
ENSURE(properlyInitialized(), "constructor must end in properlyInitialized state");
```

2.6.2.2 FlightPlan::~~FlightPlan ()

Destructor

2.6.3 Member Function Documentation

2.6.3.1 bool FlightPlan::complete () const

Checks if all the data members were initialized

REQUIRE(this->properlyInitialized(), "Flightplan wasn't initialized when calling complete");

Returns

: boolean indicating if all members are initialized.

2.6.3.2 const string & FlightPlan::getDestination () const

Getters and setters for the fields of the class. For all: REQUIRE(properlyInitialized(), "Flightplan wasn't properly initialized when calling getter/setter.");

2.6.3.3 EEvent FlightPlan::getEvent (Time time)

Returns the event at the given time

REQUIRE(this->properlyInitialized(), "Flightplan wasn't initialized when calling getEvent");

Parameters

| | |
|-------------|----------------------|
| <i>time</i> | time to get event at |
|-------------|----------------------|

Returns

event at this time

2.6.3.4 bool FlightPlan::properlyInitialized () const

Checks if the object is properly initialized

Returns

: boolean indicating if properly initialized

2.6.3.5 void FlightPlan::setArrival (int arrival)

Setter for arrival time

REQUIRE(this->properlyInitialized(), "Flightplan wasn't initialized when calling setArrival");

REQUIRE(arrival >= 0 && arrival < 60, "Arrival has to be between 0 and 60");

Parameters

| | |
|----------------|---------------------|
| <i>arrival</i> | arrival time to set |
|----------------|---------------------|

2.6.3.6 void FlightPlan::setDeparture (int *departure*)

Setter for departure time

REQUIRE(this->[properlyInitialized\(\)](#), "Flightplan wasn't initialized when calling setDeparture");

REQUIRE(departure >= 0 && departure < 60, "Departure has to be between 0 and 60");

Parameters

| | |
|------------------|-----------------------|
| <i>departure</i> | departure time to set |
|------------------|-----------------------|

2.6.3.7 void FlightPlan::setInterval (int *interval*)

Setter for interval

REQUIRE(this->[properlyInitialized\(\)](#), "Flightplan wasn't initialized when calling setInterval");

REQUIRE(interval > 0, "Interval has to be at least 1");

Parameters

| | |
|-----------------|-----------------|
| <i>interval</i> | interval to set |
|-----------------|-----------------|

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

- /home/max/CLionProjects/ProjectVliegveld/headers/FlightPlan.h
- /home/max/CLionProjects/ProjectVliegveld/src/FlightPlan.cpp

2.7 Graphics Class Reference

```
#include <Graphics.h>
```

Public Member Functions

- [Graphics](#) ([Airport](#) *airport)
- void [addElement](#) ([Airplane](#) *airplane)
- void [addElement](#) ([Runway](#) *runway)
- std::string [generateINI](#) (double x, double y, double z, int size=3000) const
- bool [properlyInitialized](#) () const

2.7.1 Detailed Description

Class that generates a string in valid .ini format.

This string can be used with the [Graphics](#) Engine to generate an image of all the added elements and thus the airport.

2.7.2 Constructor & Destructor Documentation

2.7.2.1 Graphics::Graphics (Airport * airport)

Constructor. Adds the figures for the gates.

ENSURE([properlyInitialized\(\)](#), "Graphics object was not properly constructed");

Parameters

| | |
|----------------|---------------------------|
| <i>airport</i> | airport in the simulation |
|----------------|---------------------------|

2.7.3 Member Function Documentation

2.7.3.1 void Graphics::addElement (Airplane * airplane)

Adds the figures for an airplane.

The position is calculated with the status and position of the airplane.

REQUIRE([properlyInitialized\(\)](#), "Graphics was not properly initialized when calling addElement(airplane)");

REQUIRE(runway != NULL, "Element can't be NULL when calling addElement");

Parameters

| | |
|-----------------|--------------------------|
| <i>airplane</i> | the airplane to be added |
|-----------------|--------------------------|

2.7.3.2 void Graphics::addElement (Runway * runway)

Adds the figures for a runway.

The position is calculated by the number of runways already added

REQUIRE([properlyInitialized\(\)](#), "Graphics was not properly initialized when calling addElement(runway)");

REQUIRE(airplane != NULL, "Element can't be NULL when calling addElement");

Parameters

| | |
|---------------|------------------------|
| <i>runway</i> | the runway to be added |
|---------------|------------------------|

2.7.3.3 std::string Graphics::generateINI (double x, double y, double z, int size = 3000) const

Generates the ini file of all the elements in the figures vector.

REQUIRE([properlyInitialized\(\)](#), "Graphics was not properly initialized when calling generateINI");

REQUIRE(size > 0, "Size can't be negative");

Parameters

| | |
|----------|------------------------|
| <i>x</i> | eye point x coordinate |
| <i>y</i> | eye point y coordinate |
| <i>z</i> | eye point z coordinate |

Returns

string in valid ini format

2.7.3.4 bool Graphics::properlyInitialized () const

Checks if the object is properly initialized

Returns

: Boolean indicating if properly initialized or not.

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

- /home/max/CLionProjects/ProjectVliegveld/headers/Graphics.h
- /home/max/CLionProjects/ProjectVliegveld/src/Graphics.cpp

2.8 Input Class Reference

: Class that reads input for the simulation

```
#include <Input.h>
```

Public Member Functions

- [Input](#) ()
- void [read](#) (const std::string &filename, std::ostream &errorLog=std::cerr)
- void [addAirport](#) ([Airport](#) *airport)
- void [addRunway](#) ([Runway](#) *runway)
- void [addFlightPlan](#) ([FlightPlan](#) *flightPlan)
- [Airport](#) * [findAirportByIATA](#) (const std::string &iata) const
- std::vector< [Airport](#) * > [getAirports](#) () const
- std::vector< [FlightPlan](#) * > [getFlightPlans](#) () const
- bool [properlyInitialized](#) () const

Static Public Member Functions

- static bool [isNumber](#) (const std::string &input)

2.8.1 Detailed Description

: Class that reads input for the simulation

2.8.2 Constructor & Destructor Documentation

2.8.2.1 Input::Input ()

Default constructor

ENSURE([properlyInitialized\(\)](#), "constructor must end in properlyInitialized state");

2.8.3 Member Function Documentation

2.8.3.1 void Input::addAirport (Airport * airport)

Adds an airport if all data members are initialized.

REQUIRE(this->[properlyInitialized\(\)](#), "Input wasn't initialized when calling addAirport");
 REQUIRE(airport->complete(), "Airport has to be completely initialized to add it to the simulation");
 ENSURE([getAirports\(\)](#).back() == airport, "Airplane was not added to simulation.");

Parameters

| | |
|----------------|-------------------------|
| <i>airport</i> | the airport to be added |
|----------------|-------------------------|

2.8.3.2 void Input::addFlightPlan (FlightPlan * flightPlan)

Adds a flight plan if all data members are initialized.

REQUIRE(this->[properlyInitialized\(\)](#), "Input wasn't initialized when calling addFlightplan");
 REQUIRE(flightPlan->complete(), "FlightPlan has to be completely initialized to add it to the simulation");
 ENSURE([getFlightPlans\(\)](#).back() == flightPlan, "FlightPlan was not added to simulation.");

Parameters

| | |
|-------------------|-----------------------------|
| <i>flightPlan</i> | the flight plan to be added |
|-------------------|-----------------------------|

2.8.3.3 void Input::addRunway (Runway * runway)

Adds a runway if all data members are initialized.

REQUIRE(this->[properlyInitialized\(\)](#), "Input wasn't initialized when calling addRunway");
 REQUIRE(runway->complete(), "Runway has to be completely initialized to add it to the simulation");
 ENSURE(runway->getAirport()->getRunways().back() == runway, "Runway was not added to the airport");

Parameters

| | |
|---------------|------------------------|
| <i>runway</i> | the runway to be added |
|---------------|------------------------|

2.8.3.4 Airport * Input::findAirportByIATA (const std::string & iata) const

Finds an airport with a specific IATA.

Returns NULL if not found.

REQUIRE(this->[properlyInitialized\(\)](#), "Input wasn't initialized when calling findAirportByIATA");

Parameters

| | |
|-------------|--------------------------------|
| <i>iata</i> | the iata of the wanted airport |
|-------------|--------------------------------|

Returns

the airport if found

2.8.3.5 `vector< Airport * > Input::getAirports () const`

Getter for the airports in the simulation

REQUIRE(this->[properlyInitialized\(\)](#), "Input wasn't initialized when calling getAirports");

Returns

vec of all airports

2.8.3.6 `vector< FlightPlan * > Input::getFlightPlans () const`

Getter for the flight plans in the simulation

REQUIRE(this->[properlyInitialized\(\)](#), "Input wasn't initialized when calling getFlightPlans");

Returns

vec of all flight plans

2.8.3.7 `bool Input::isNumber (const std::string & input) [static]`

Checks if a given std::string is a valid unsigned int

REQUIRE(this->[properlyInitialized\(\)](#), "Input wasn't initialized when calling isNumber");

Parameters

| | |
|--------------|--------------------------------|
| <i>input</i> | the std::string to be analyzed |
|--------------|--------------------------------|

Returns

boolean isNumber

2.8.3.8 `bool Input::properlyInitialized () const`

Checks if the object is properly initialized

2.8.3.9 void Input::read (const std::string & *filename*, std::ostream & *errorLog* = std::cerr)

Reads the given file and stores the information

REQUIRE(TiXmlDocument::LoadFile(filename.c_str()), "Couldn't open \$filename.");

REQUIRE(this->properlyInitialized(), "Input wasn't initialized when calling read");

Parameters

| | |
|-----------------|-----------------------------------------------|
| <i>filename</i> | name of the file with input |
| <i>errorLog</i> | output stream where errors will be written to |

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

- /home/max/CLionProjects/ProjectVliegveld/headers/Input.h
- /home/max/CLionProjects/ProjectVliegveld/src/Input.cpp

2.9 Runway Class Reference

```
#include <Runway.h>
```

Public Member Functions

- [Runway](#) ()
- bool [complete](#) () const
- bool [properlyInitialized](#) () const
- bool [validForAirplane](#) ([Airplane](#) *plane) const
- ERunwayType [getType](#) () const
- void [setType](#) (ERunwayType type)
- int [getLength](#) () const
- void [setLength](#) (int length)
- const std::string & [getName](#) () const
- void [setName](#) (const std::string &fName)
- bool [isFree](#) () const
- void [setFree](#) (bool free)
- std::string [getTaxiPoint](#) () const
- void [setTaxiPoint](#) (const std::string &)
- [Airport](#) * [getAirport](#) () const
- void [setAirport](#) ([Airport](#) *fAirport)

2.9.1 Detailed Description

Class that represents a runway in an airport

2.9.2 Constructor & Destructor Documentation

2.9.2.1 Runway::Runway ()

Constructor for the [Runway](#) class.

ENSURE([properlyInitialized](#)(), "Runway wasn't properly initialized after constructing.");

2.9.3 Member Function Documentation

2.9.3.1 `bool Runway::complete () const`

Checks if all the data members were initialized

REQUIRE([properlyInitialized\(\)](#), "Runway wasn't properly initialized when calling complete.");

Returns

: Boolean indicating if all members were initialized

2.9.3.2 `ERunwayType Runway::getType () const`

Getters and setters for the fields of the class. For all: REQUIRE([properlyInitialized\(\)](#), "Runway wasn't properly initialized when calling getter/setter."); For setters; ENSURE(getField == value, "Field wasn't set properly"); where getField is specific for the member

2.9.3.3 `bool Runway::properlyInitialized () const`

Checks if the object is properly initialized

Returns

: Boolean indicating if properly initialized or not.

2.9.3.4 `bool Runway::validForAirplane (Airplane * plane) const`

Check if this runway is valid for the provided airplane.

REQUIRE([properlyInitialized\(\)](#), "Runway wasn't properly initialized when calling validForAirplane.");

REQUIRE(plane != NULL, "Plane object does not exist.");

Parameters

| | |
|--------------|-------------------------------------------------|
| <i>plane</i> | Airplane to check validity for. |
|--------------|-------------------------------------------------|

Returns

: Boolean indicating if valid or not.

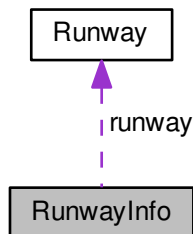
The documentation for this class was generated from the following files:

- /home/max/CLionProjects/ProjectVliegveld/headers/Runway.h
- /home/max/CLionProjects/ProjectVliegveld/src/Runway.cpp

2.10 RunwayInfo Struct Reference

```
#include <Graphics.h>
```

Collaboration diagram for RunwayInfo:



Public Attributes

- [Runway](#) * runway
- int [arrivingPlanes](#)
- int [departingPlanes](#)

2.10.1 Detailed Description

Struct containing info about a runway

2.10.2 Member Data Documentation

2.10.2.1 int RunwayInfo::arrivingPlanes

Arriving planes at taxipoint of runway

2.10.2.2 int RunwayInfo::departingPlanes

Departing planes at taxipoint of runway

2.10.2.3 Runway* RunwayInfo::runway

The runway

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

- /home/max/CLionProjects/ProjectVliegveld/headers/Graphics.h

2.11 System Class Reference

: Main class, controls the simulation.

```
#include <System.h>
```

Public Member Functions

- [System](#) (const [Input](#) &input, std::ostream &atc, const [Time](#) &end)
- [~System](#) ()
- void [run](#) (std::ostream &log, const std::string &impressionName="../output/impressions/impression", const std::string &iniName="../output/ini/graphics")
- void [info](#) (std::ostream &out)
- void [generateImages](#) ([Time](#) start, [Time](#) end)
- bool [simulationFinished](#) () const
- [Airport](#) * [getAirport](#) () const
- [ATC](#) * [getATC](#) () const
- std::vector< [FlightPlan](#) * > [getFlightPlans](#) () const
- bool [properlyInitialized](#) () const

2.11.1 Detailed Description

: Main class, controls the simulation.

2.11.2 Constructor & Destructor Documentation

2.11.2.1 System::System (const [Input](#) & *input*, std::ostream & *atc*, const [Time](#) & *end*)

Constructor

REQUIRE(input.getAirports().empty(), "There has to be an airport in the input to start the simulation");
ENSURE([properlyInitialized](#)(), "constructor must end in properlyInitialized state");

Parameters

| | |
|--------------|--------------------------------------------------|
| <i>input</i> | the input of the simulation |
| <i>atc</i> | the stream where atc messages will be written to |
| <i>end</i> | the ending time of the simulation |

2.11.2.2 System::~~System ()

Destructor

2.11.3 Member Function Documentation

2.11.3.1 void System::generateImages (Time start, Time end)

Generates the images from the start time until the end time, with the use of the generated ini files and the graphics engine.

REQUIRE(this->properlyInitialized(), "System was't initialized when calling generateImages");

Parameters

| | |
|--------------|----------------------------------|
| <i>start</i> | time of first image |
| <i>end</i> | time of last image, not included |

2.11.3.2 Airport * System::getAirport () const

Getter for the airport in the simulation

REQUIRE(this->properlyInitialized(), "System was't initialized when calling getAirport");

Returns

: airport in the simulation

2.11.3.3 ATC * System::getATC () const

Getter for the air traffic control in the simulation

REQUIRE(this->properlyInitialized(), "System was't initialized when calling getATC");

Returns

: [ATC](#) in the simulation

2.11.3.4 vector< FlightPlan * > System::getFlightPlans () const

Getter for the flight plans in the simulation

REQUIRE(this->properlyInitialized(), "System was't initialized when calling getFlightPlans);

Returns

: vector of flightplans that are in the simulation

2.11.3.5 void System::info (std::ostream & out)

Logs information of the airports and airplanes to a text file

REQUIRE(this->properlyInitialized(), "System was't initialized when calling info");

REQUIRE(fAirport != NULL, "No airport in the simulation");

Parameters

| | |
|------------|-----------------------------------------------|
| <i>out</i> | the ostream where the info will be written to |
|------------|-----------------------------------------------|

2.11.3.6 `bool System::properlyInitialized () const`

Checks if the object is properly initialized

Returns

: boolean indicating if object is properly initialized.

```
2.11.3.7 void System::run ( std::ostream & log, const std::string & impressionName =
    "../output/impressions/impression", const std::string & iniName =
    "../output/ini/graphics" )
```

Runs the complete simulation

```

REQUIRE(this->properlyInitialized(), "System was't initialized when calling run");
REQUIRE(getAirport() != NULL, "No airport in the simulation.");
REQUIRE(!simulationFinished(), "Simulation is already finished");
ENSURE(simulationFinished(), "Simulation is not finished yet, error occurred");
```

Parameters

| | |
|-----------------------|-------------------------------------------------------|
| <i>log</i> | ostream where all the log messages will be written to |
| <i>impressionName</i> | basename of the files for the impressions |
| <i>iniName</i> | basename of the files for the ini files |

2.11.3.8 `bool System::simulationFinished () const`

Checks if the simulation has ended, i.e. specified end time has been reached

```
REQUIRE(this->properlyInitialized(), "System was't initialized when calling simulationFinished");
```

Returns

: boolean indicating if the simulation is finished.

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

- /home/max/CLionProjects/ProjectVliegveld/headers/System.h
- /home/max/CLionProjects/ProjectVliegveld/src/System.cpp

2.12 Time Class Reference

```
#include <Time.h>
```

Public Member Functions

- [Time](#) (int hour=12, int minute=0)
- [Time](#) (const [Time](#) &time)
- std::string [formatted](#) () const
- void [advance](#) (int minutes=1)
- void [setMinute](#) (int minute)
- int [getMinute](#) () const
- void [setHour](#) (int hour)
- int [getHour](#) () const
- bool [properlyInitialized](#) () const
- bool [operator==](#) (const [Time](#) &) const
- bool [operator<](#) (const [Time](#) &) const
- [Time](#) & [operator=](#) (const [Time](#) &time)

2.12.1 Detailed Description

Class that represents the time

2.12.2 Constructor & Destructor Documentation

2.12.2.1 [Time::Time](#) (int *hour* = 12, int *minute* = 0)

Constructor, sets the time

Defaults to 12:00, which is the starting point of the simulation

REQUIRE(minute < 60 && minute >= 0, "Minute has to be between 0 and 60");

REQUIRE(hour < 24 && hour >= 0, "Hour has to be between 0 and 24");

ENSURE([properlyInitialized\(\)](#), "Time wasn't properly initialized after constructing.");

Parameters

| | |
|---------------|------------|
| <i>hour</i> | the hour |
| <i>minute</i> | the minute |

2.12.2.2 [Time::Time](#) (const [Time](#) & *time*)

Copy constructor

ENSURE([properlyInitialized\(\)](#), "Time wasn't properly initialized after constructing.");

Parameters

| | |
|-------------|---------------------|
| <i>time</i> | object to be copied |
|-------------|---------------------|

2.12.3 Member Function Documentation

2.12.3.1 void Time::advance (int *minutes* = 1)

Advances the time by an amount of minutes

REQUIRE([properlyInitialized\(\)](#), "Time wasn't properlyInitialized when calling advance");
 REQUIRE(minutes >= 0, "Advancing by a negative amount of minutes is not possible");

Parameters

| | |
|----------------|------------------------------|
| <i>minutes</i> | amount of minutes to advance |
|----------------|------------------------------|

2.12.3.2 string Time::formatted () const

Return the time in a formatted style like such: "13:45"

REQUIRE([properlyInitialized\(\)](#), "Time wasn't properlyInitialized when calling formatted");

Returns

string of time

2.12.3.3 int Time::getHour () const

Getter for the hour

REQUIRE([properlyInitialized\(\)](#), "Time wasn't properlyInitialized when calling getHour");

Returns

: hour

2.12.3.4 int Time::getMinute () const

Getter for the minute

REQUIRE([properlyInitialized\(\)](#), "Time wasn't properlyInitialized when calling getMinute");

Returns

: minute

2.12.3.5 bool Time::operator< (const Time & *time*) const

Operator< overloaded, returns false if comparing with 00:00

2.12.3.6 Time & Time::operator= (const Time & *time*)

Assignment operator

Parameters

| | |
|-------------|-----------------|
| <i>time</i> | rhs of operator |
|-------------|-----------------|

Returns

reference to this

2.12.3.7 `bool Time::operator==(const Time & time) const`

Operator== overloaded

2.12.3.8 `bool Time::properlyInitialized () const`

Checks if the object is properly initialized

Returns

: Boolean indicating if properly initialized or not.

2.12.3.9 `void Time::setHour (int hour)`

Setter for the hour.

```
REQUIRE(hour < 24 && hour >= 0, "Hour has to be between 0 and 24");  
REQUIRE(properlyInitialized\(\), "Time wasn't properlyInitialized when calling setHour");
```

Parameters

| | |
|-------------|-------------|
| <i>hour</i> | hour to set |
|-------------|-------------|

2.12.3.10 `void Time::setMinute (int minute)`

Setter for the minute.

```
REQUIRE(minute < 60 && minute >= 0, "Minute has to be between 0 and 60");  
REQUIRE(properlyInitialized\(\), "Time wasn't properlyInitialized when calling setMinute");
```

Parameters

| | |
|---------------|---------------|
| <i>minute</i> | minute to set |
|---------------|---------------|

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

- /home/max/CLionProjects/ProjectVliegveld/headers/Time.h
- /home/max/CLionProjects/ProjectVliegveld/src/Time.cpp

Index

- ~ATC
 - ATC, [16](#)
- ~Airport
 - Airport, [11](#)
- ~FlightPlan
 - FlightPlan, [22](#)
- ~System
 - System, [32](#)
- ATCRequest, [21](#)
 - ATCRequest, [21](#)
 - fAirplane, [21](#)
 - fTime, [21](#)
- ATC, [15](#)
 - ~ATC, [16](#)
 - ATC, [15](#)
 - doHeartbeat, [16](#)
 - formatMessage, [16](#)
 - getNextRequest, [16](#)
 - getQueue, [16](#)
 - getQueueSize, [16](#)
 - getSquawk, [16](#)
 - getTime, [17](#)
 - processApproach, [17](#)
 - processDescend, [17](#)
 - processEmergency, [17](#)
 - processIFRClearance, [18](#)
 - processPushback, [18](#)
 - processTakeOff, [18](#)
 - processTakeOffRunway, [18](#)
 - processTaxiArrival, [19](#)
 - processTaxiInitialise, [19](#)
 - processTaxiInstruction, [19](#)
 - processUrgentEmergency, [19](#)
 - properlyInitialized, [20](#)
 - sendMessage, [20](#)
 - sendRequest, [20](#)
 - setTime, [20](#)
- addAirport
 - Input, [27](#)
- addElement
 - Graphics, [25](#)
- addFlightPlan
 - Input, [27](#)
- addRunway
 - Airport, [11](#)
 - Input, [27](#)
- advance
 - Time, [35](#)
- Airplane, [4](#)
 - approach, [4](#)
 - ascend, [5](#)
 - atRunway, [5](#)
 - checkFuel, [5](#)
 - circle, [5](#)
 - complete, [5](#)
 - crossArrival, [6](#)
 - deboard, [6](#)
 - decreaseAltitude, [6](#)
 - decreaseTimeRemaining, [6](#)
 - descend, [6](#)
 - getFuelCost, [7](#)
 - getSize, [7](#)
 - increaseAltitude, [7](#)
 - onRunway, [7](#)
 - performNextStep, [7](#)
 - prepare, [7](#)
 - properlyInitialized, [8](#)
 - pushback, [8](#)
 - setAltitude, [8](#)
 - setCurFuel, [8](#)
 - setFuel, [8](#)
 - setGateID, [8](#)
 - setPassengers, [9](#)
 - setSquawk, [9](#)
 - setTimeRemaining, [9](#)
 - taxiArrival, [9](#)
 - taxiDepartureCross, [9](#)
 - taxiDepartureStart, [9](#)
 - taxiDepartureStep, [10](#)
 - technicalCheck, [10](#)
- Airport, [10](#)
 - ~Airport, [11](#)
 - addRunway, [11](#)
 - Airport, [11](#)
 - amountOfRunways, [12](#)
 - complete, [12](#)
 - drawImpression, [12](#)
 - getFreeGate, [12](#)
 - getFreeRunway, [12](#)
 - getName, [13](#)
 - getNextRunway, [13](#)
 - getRunway, [13](#)
 - graphicsINI, [13](#)
 - initGates, [14](#)
 - properlyInitialized, [14](#)
 - restoreGate, [14](#)
 - setGates, [14](#)

- amountOfRunways
 - Airport, [12](#)
- approach
 - Airplane, [4](#)
- arrivingPlanes
 - RunwayInfo, [31](#)
- ascend
 - Airplane, [5](#)
- atRunway
 - Airplane, [5](#)
- checkFuel
 - Airplane, [5](#)
- circle
 - Airplane, [5](#)
- Comparator, [22](#)
- complete
 - Airplane, [5](#)
 - Airport, [12](#)
 - FlightPlan, [23](#)
 - Runway, [30](#)
- crossArrival
 - Airplane, [6](#)
- deboard
 - Airplane, [6](#)
- decreaseAltitude
 - Airplane, [6](#)
- decreaseTimeRemaining
 - Airplane, [6](#)
- departingPlanes
 - RunwayInfo, [31](#)
- descend
 - Airplane, [6](#)
- doHeartbeat
 - ATC, [16](#)
- drawImpression
 - Airport, [12](#)
- fAirplane
 - ATCRequest, [21](#)
- fTime
 - ATCRequest, [21](#)
- findAirportByIATA
 - Input, [27](#)
- FlightPlan, [22](#)
 - ~FlightPlan, [22](#)
 - complete, [23](#)
 - FlightPlan, [22](#)
 - getDestination, [23](#)
 - getEvent, [23](#)
 - properlyInitialized, [23](#)
 - setArrival, [23](#)
 - setDeparture, [24](#)
 - setInterval, [24](#)
- formatMessage
 - ATC, [16](#)
- formatted
 - Time, [36](#)
- generateINI
 - Graphics, [25](#)
- generateImages
 - System, [32](#)
- getATC
 - System, [33](#)
- getAirport
 - System, [33](#)
- getAirports
 - Input, [28](#)
- getDestination
 - FlightPlan, [23](#)
- getEvent
 - FlightPlan, [23](#)
- getFlightPlans
 - Input, [28](#)
 - System, [33](#)
- getFreeGate
 - Airport, [12](#)
- getFreeRunway
 - Airport, [12](#)
- getFuelCost
 - Airplane, [7](#)
- getHour
 - Time, [36](#)
- getMinute
 - Time, [36](#)
- getName
 - Airport, [13](#)
- getNextRequest
 - ATC, [16](#)
- getNextRunway
 - Airport, [13](#)
- getQueue
 - ATC, [16](#)
- getQueueSize
 - ATC, [16](#)
- getRunway
 - Airport, [13](#)
- getSize
 - Airplane, [7](#)
- getSquawk
 - ATC, [16](#)
- getTime
 - ATC, [17](#)
- getType
 - Runway, [30](#)
- Graphics, [24](#)
 - addElement, [25](#)
 - generateINI, [25](#)
 - Graphics, [25](#)
 - properlyInitialized, [26](#)
- graphicsINI
 - Airport, [13](#)
- increaseAltitude
 - Airplane, [7](#)
- info
 - System, [33](#)

- initGates
 - Airport, [14](#)
- Input, [26](#)
 - addAirport, [27](#)
 - addFlightPlan, [27](#)
 - addRunway, [27](#)
 - findAirportByIATA, [27](#)
 - getAirports, [28](#)
 - getFlightPlans, [28](#)
 - Input, [27](#)
 - isNumber, [28](#)
 - properlyInitialized, [28](#)
 - read, [28](#)
- isNumber
 - Input, [28](#)
- onRunway
 - Airplane, [7](#)
- operator<
 - Time, [36](#)
- operator=
 - Time, [36](#)
- operator==
 - Time, [37](#)
- performNextStep
 - Airplane, [7](#)
- prepare
 - Airplane, [7](#)
- processApproach
 - ATC, [17](#)
- processDescend
 - ATC, [17](#)
- processEmergency
 - ATC, [17](#)
- processIFRClearance
 - ATC, [18](#)
- processPushback
 - ATC, [18](#)
- processTakeOff
 - ATC, [18](#)
- processTakeOffRunway
 - ATC, [18](#)
- processTaxiArrival
 - ATC, [19](#)
- processTaxiInitialise
 - ATC, [19](#)
- processTaxiInstruction
 - ATC, [19](#)
- processUrgentEmergency
 - ATC, [19](#)
- properlyInitialized
 - ATC, [20](#)
 - Airplane, [8](#)
 - Airport, [14](#)
 - FlightPlan, [23](#)
 - Graphics, [26](#)
 - Input, [28](#)
 - Runway, [30](#)
 - System, [34](#)
 - Time, [37](#)
- pushback
 - Airplane, [8](#)
- read
 - Input, [28](#)
- restoreGate
 - Airport, [14](#)
- run
 - System, [34](#)
- Runway, [29](#)
 - complete, [30](#)
 - getType, [30](#)
 - properlyInitialized, [30](#)
 - Runway, [29](#)
 - validForAirplane, [30](#)
- runway
 - RunwayInfo, [31](#)
- RunwayInfo, [31](#)
 - arrivingPlanes, [31](#)
 - departingPlanes, [31](#)
 - runway, [31](#)
- sendMessage
 - ATC, [20](#)
- sendRequest
 - ATC, [20](#)
- setAltitude
 - Airplane, [8](#)
- setArrival
 - FlightPlan, [23](#)
- setCurFuel
 - Airplane, [8](#)
- setDeparture
 - FlightPlan, [24](#)
- setFuel
 - Airplane, [8](#)
- setGateID
 - Airplane, [8](#)
- setGates
 - Airport, [14](#)
- setHour
 - Time, [37](#)
- setInterval
 - FlightPlan, [24](#)
- setMinute
 - Time, [37](#)
- setPassengers
 - Airplane, [9](#)
- setSquawk
 - Airplane, [9](#)
- setTime
 - ATC, [20](#)
- setTimeRemaining
 - Airplane, [9](#)
- simulationFinished
 - System, [34](#)
- System, [32](#)

- ~System, [32](#)
- generateImages, [32](#)
- getATC, [33](#)
- getAirport, [33](#)
- getFlightPlans, [33](#)
- info, [33](#)
- properlyInitialized, [34](#)
- run, [34](#)
- simulationFinished, [34](#)
- System, [32](#)

taxiArrival

- Airplane, [9](#)

taxiDepartureCross

- Airplane, [9](#)

taxiDepartureStart

- Airplane, [9](#)

taxiDepartureStep

- Airplane, [10](#)

technicalCheck

- Airplane, [10](#)

Time, [34](#)

- advance, [35](#)
- formatted, [36](#)
- getHour, [36](#)
- getMinute, [36](#)
- operator<, [36](#)
- operator=, [36](#)
- operator==, [37](#)
- properlyInitialized, [37](#)
- setHour, [37](#)
- setMinute, [37](#)
- Time, [35](#)

validForAirplane

- Runway, [30](#)