AirportSimulation

Generated by Doxygen 1.8.11

Contents

Chapter 1

Class Index

1.1 Class List

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

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 was't initialized when calling approach");

Parameters

log	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

log	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

log	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

log	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 was't initialized when calling circle");

Parameters

log	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

log	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

log	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

log	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 was'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

log	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

log	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

log	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

log	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 was'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 was'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 was'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 was'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 was'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 was'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 was'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 was't initialized when calling taxiArrival");

Parameters

log	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

log	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

log	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

log	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 was't initialized when calling technicalCheck");

Parameters

log	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

runway	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

time	time of the impression.
plans	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

plane	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

airplane	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

plans	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

id	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

gates	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

stream	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

time	Time of message.
source	Sender of message.
message	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

airplane	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 was'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

airplane	the sender of the request
time	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

airplane	the sender of the request
time	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

airplane	the sender of the request
time	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

airplane	the sender of the request
time	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

airplane	the sender of the request
time	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

airplane	the sender of the request
time	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

airplane	the sender of the request
time	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

airplane	the sender of the request
time	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

airplane	the sender of the request
time	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

airplane	the sender of the request
time	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

airplane	the sender of the request
time	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

message	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

time	Time of the request.
source	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

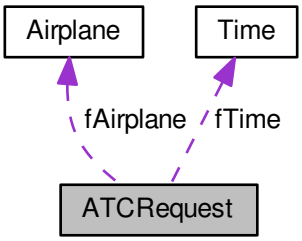
time	time to set

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 was'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 was't initialized when calling getEvent");

Parameters

time	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 was't initialized when calling setArrival");
REQUIRE(arrival >= 0 && arrival < 60, "Arrival has to be between 0 and 60");

Parameters

arrival	arrival time to set

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

Setter for departure time
REQUIRE(this->properlyInitialized(), "Flightplan was't initialized when calling setDeparture");
REQUIRE(departure >= 0 && departure < 60, "Departure has to be between 0 and 60");

Parameters

departure	departure time to set

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

Setter for interval
REQUIRE(this->properlyInitialized(), "Flightplan was't initialized when calling setInterval");
REQUIRE(interval > 0, "Interval has to be at least 1");

Parameters

interval	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

airport	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

airplane	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

runway	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

x	eye point x coordinate
y	eye point y coordinate
z	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 was'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

airport	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 was'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

flightPlan	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 was'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

runway	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 was't initialized when calling findAirportByIATA");

Parameters

iata	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 was'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 was'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 was't initialized when calling isNumber");

Parameters

input	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 was't initialized when calling read");

Parameters

filename	name of the file with input
errorLog	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

plane	Airplane to check validity for.

Returns

: Boolean indicating if valid or not.

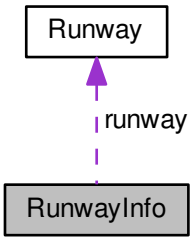
The documentation for this class was generated from the following files:

- /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

input	the input of the simulation
atc	the stream where atc messages will be written to
end	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

start	time of first image
end	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

out	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

log	ostream where all the log messages will be written to
impressionName	basename of the files for the impressions
iniName	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

hour	the hour
minute	the minute

2.12.2.2 Time::Time (const Time & *time*)

Copy constructor
ENSURE(properlyInitialized(), "Time wasn't properly initialized after constructing.");

Parameters

time	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

minutes	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

time	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

hour	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

minute	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