Simulation Module (sim)

# Class Simulation

* This class is in simulation.py
* It is a derived class. Its base class is 'QObject'. Therefore it inherits many methods that are not visible in simulation.py
* It can be used to create a sim object, which in turn contains other objects (such as the traf and stack and scnreenio.

## \_\_init\_\_(self, gui, navdb)

Goal: Initializes the simulation object. The simulation object contains variables to control the simulation, like simt (simulation time), as well as objects, like traf and stack. Therefore, the simulation object can easily 'communicate' with the methods/functions of other classes such as Traffic and CommandStack.

Inputs:

1. gui(object, instance of the Gui class)
2. navdb(object, instance of the Navdatabase class)

Ouputs: -

## moveToThread(self, target\_thread)

Goal: In BlueSky, the simulation and the gui are on separate threads. This functions specifies which thread different parts of BlueSky should be moved to. It makes a call to the moveToThread method of the base QObject class. **Best not to modify this function if you don't know anything about multi-threaded programs!**

Inputs:

1. target\_thread(object, instance of the Thread class)

Ouputs: -

## doWork(self)

Goal: This is the main method of the Simulation Class. It calls methods from ComandStack and Traffic classes (amongst other classes) in a while loop until the simulation is quit. The methods of the other classes that are called from within the doWork function are, for example, responsible for processing all stack commands, updating the states of the traffic. In this sense, this function contains the main simulation loop (even though there is another function called MainLoop in the Simulation module).

Inputs: -

Outputs: -

## stop(self)

Goal: As the name suggests, this function is called (by the MainLoop function, see below) to stop the simulation. It changes the mode of the simulation to 'end' (or mode==4) and asks the screenio object to stop the visualization.

Inputs: -

Outputs: -

## start(self)

Goal: As the name suggests, this function is called to start the simulation. It changes the mode of the simulation to 'op' (or mode==1). It is called from several places, including the *doWork* function of the Simulation class (see above), and by various BlueSky commands when they are processed by the *process* function in the CommandStack class.

Inputs: -

Outputs: -

## pause(self)

Goal: As the name suggests, this function is called to pause simulation. It changes the mode of the simulation to 'op' (or mode==2). It is called when the *hold* BlueSky command is processed by the *process* function in the CommandStack class.

Inputs: -

Outputs: -

## reset(self)

Goal: As the name suggests, this function is called to reset simulation. It changes the mode of the simulation to 'init' (or mode==0). It is called when the *IC* BlueSky command is processed by the *process* function in the CommandStack class to load a new scenario file. It also resets the simulation time (simt) to 0 and resets the traffic arrays by calling the traf.reset() function in the Traffic class.

Inputs: -

Outputs: -

## fastforward(self)

Goal: As the name suggests, this function is used to speed up the simulation, i.e., a Fast Time simulation. It is called when the *FF* BlueSky command is processed by the *process* function in the CommandStack class.

Inputs: -

Outputs: -

## datafeed(self,flag)

Goal: This function activates the feeding of external data to blueSky, such as from an ADSB antenna. It is called when the *DATAFEED* BlueSky command is processed by the *process* function in the CommandStack class.

Inputs:

1. flag(string, 'ON' or 'OFF' to turn on or off the data-feed).

Outputs: -