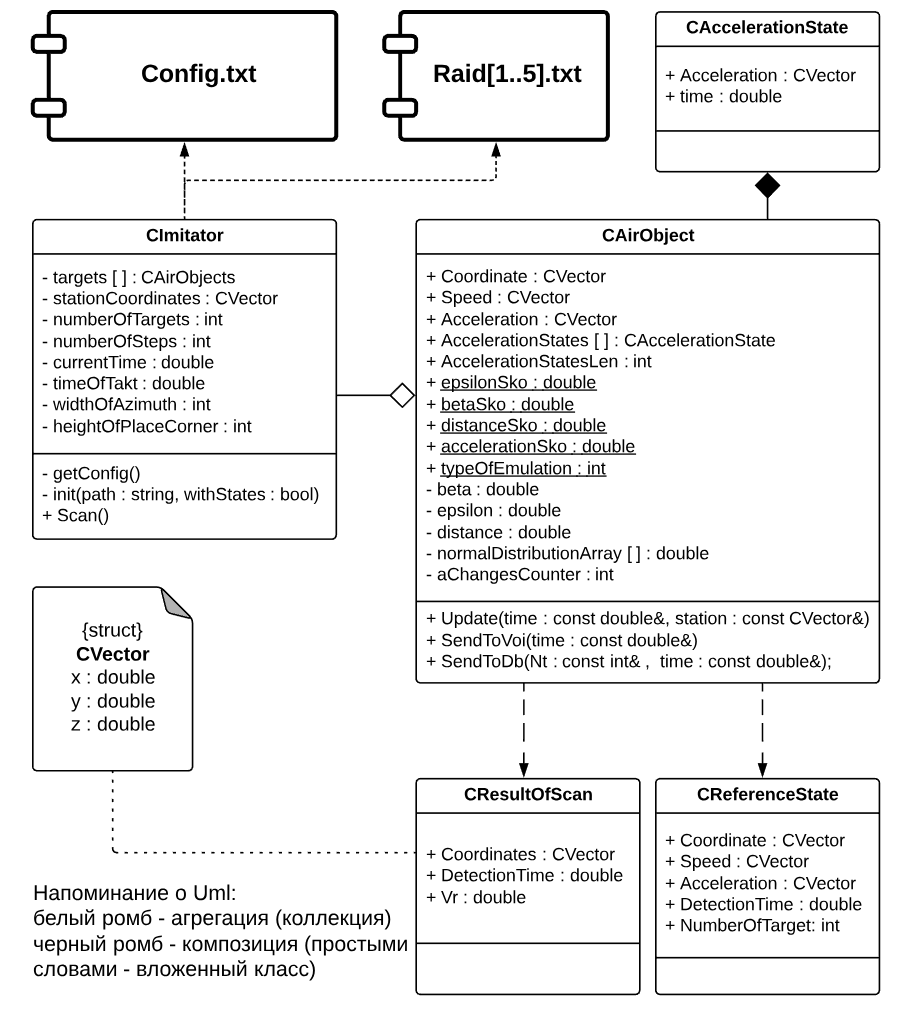
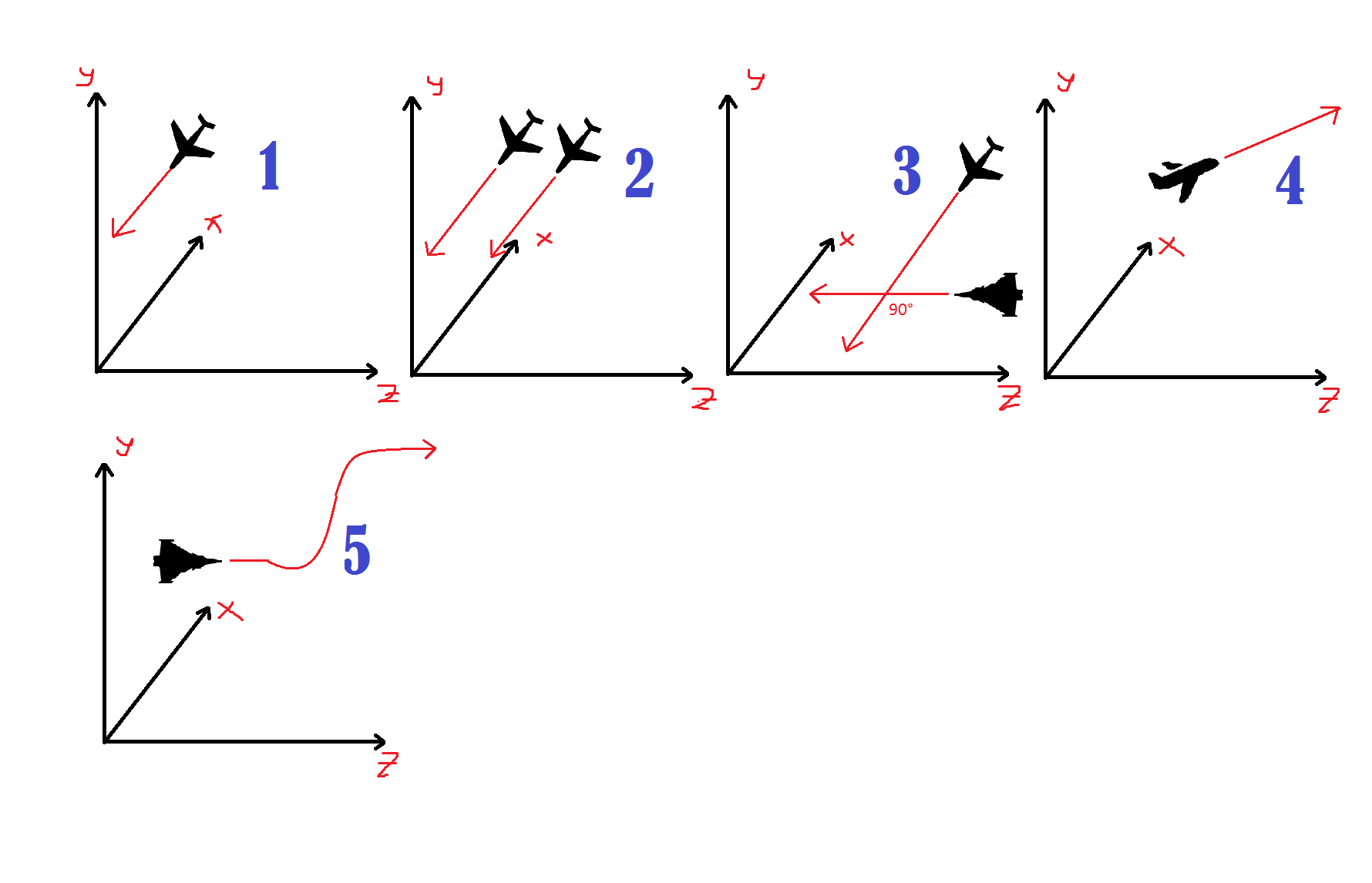
Имитатор.UML Диаграмма:



Текстовое описание:

Программа имеет один главный и несколько дополнительных конфигурационных файлов. Главный файл Config.txt предлагает определить все параметры, касающиеся работы радиолокационной станции, все необходимые среднеквадратические отклонения для шумов и тип симуляции, который определяет, каким дополнительным конфигурационным файлом будут описываться параметры воздушных целей. На данный момент таких файлов 5 (Raid1.txt, Raid2.txt …). Эти конфигурационные файлы уже содержат некоторые конкретные конфигурации, однако все они поддаются изменениям, как это сделать описано в короткой инструкции в каждом таком файле. По сути, их можно разделить на два типа – симулирующие прямолинейные движения (с 1 по 4 конфигурации) и симулирующие движения с поворотами за счет изменений параметров вектора ускорения (конфигурация 5), там вы можете для каждой цели задать не только начальные значения векторов, но и установить моменты времени, когда цель изменит направления своего движения и сами эти направления посредством установления таймера и новых значений вектора ускорения, как это делать, как уже было сказано, описано в инструкции в самом файле.  
На данный момент конфигурационные файлы симулируют следующие ситуации:



Все начинается с конструктора класса CImitator, который, выполняя метод getConfig(), парсит Config.txt, инициализируя себя, статические поля класса CAirObject, в числе которых и поле static int typeOfEmulation;, относительно которого затем выбирается конфиг файл для метода void init(char\* path, bool withStates); второй параметр которого говорит о том, какого типа конфигурационный файл (1 или 2, см. выше) для нужд парсинга. Метод первым делом считывает значение количества целей и выделяет память для массива целей соответствующего размера (поле CAirObject\* targets;).  
Далее метод рекуррентно заполняет начальными значениями векторов все цели, продолжая читать конфиг файл, и в случае 5го конфиг файла еще выделяет память и инициализирует значениями массив объектов состояний  
 CAccelerationState\* AccelerationStates;, который предназначен для хранения времени и параметров вектора ускорения для возможности каждой цели по своему непрямолинейно летать, размер этого массива для всех целей может быть разным.

Далее объект имитатор вызывает свой основной метод -> Scan(), который в двойном цикле последовательно пробегает всю область обзора станции, каждую итерация выполняется за 0,001 сек по стандарту (можно изменить в config.txt), назовем это время тактом. На каждом такте все цели вызывают у себя метод Update(const double time, const double curTime, const CVector& station);, в котором происходит пересчет их координат, скоростей, относительно которых высчитываются параметры эпсилон, бета и дистанция до цели, по первым двум в дальнейшем мы проверяем (уже в методе Scan), не смотрит ли в это место сейчас луч станции, и если смотрит – выполняем методы SendToVoi(const double curTime); и SendToDb(const int numTarget, const double curTime);,которые формируют объекты классов CResultOfScan и CReferenceState соответственно, объекты так же соответственно будут отсылаться на вторичную обработку и в базу данных, однако в базу данных пойдет и то и то. CReferenceState формирует пакет эталонных данных, а перед инстанцированием объекта CResultOfScan на эпсилон, бета и дистанцию до цели накладываются шумы, размер которых определяется среднеквадратическими отклонениями нормального распределения, которые можно по желанию изменить в Config.txt., после чего по ним уже вычисляются отправляемые координаты. Так же на ВОИ вместе с целью передается время обнаружения цели и в дальнейшем будет отправляться радиальная скорость. На этом заканчивается метод Scan(), однако он рекурсивно вызывает себя пока не истечет время моделирования, опять же задаваемое в конфиг файле.

Теперь вернемся к методу Update(const double time, const double curTime, const CVector& station); , который ежетактно вызывают все цели, я не закончил его для удобства понимания, однако на формировании эпсилон, бета и дистанции до цели его работа не заканчивается. Во-первых, если среднеквадратическое отклонение для шума ускорения задано, как не нулевое, метод будет так же ежетактно накладывать шумы на ускорение, во-вторых, если имитатор работает с конфигурацией с переменным ускорением (сейчас это конфигурация 5), будет осуществляться проверка, не зашло ли текущее время моделирования за таймер очередного состояния ускорения, и если зашло, ускорение обновляется и проверка начинает идти по таймеру уже следующего состояния, если оно есть.  
В программе часто фигурирует структура CVector, которая создана для удобства и хранит в себе три координаты (x,y,z).  
Кажется, рассказал все необходимое.