# СПЕЦИАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МОДЕЛИРОВАНИЯ БОЕВОЙ ВОЗДУШНОЙ ОБСТАНОВКИ И РЕАГИРОВАНИЯ ЗЕНИТНОЙ РАКЕТНОЙ СИСТЕМЫ «ШУРШУНЧИК»

## Руководство программиста

**Листов 18**

**2024**

# АННОТАЦИЯ

В данном программном документе приведено руководство программиста по настройке и использованию специального программного обеспечения моделирования боевой воздушной обстановки и реагирования зенитной ракетной системы «Шуршунчик».

В данном программном документе, в разделе «Назначение и условия применения программы» указаны назначение и функции, выполняемые программой, условия, необходимые для выполнения программы.

В разделе «Характеристика программы» приведено описание основных характеристик и особенностей программы.

В разделе «Обращение к программе» приведено описание процедур вызова программы

В данном программном документе, в разделе «Входные и выходные данные» приведено описание организации используемой входной и выходной информации.

В разделе «Сообщения» указаны тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы, описание их содержания и действий, которые необходимо предпринять по этим сообщениям.

Оформление программного документа «Руководство программиста» произведено по требованиям ЕСПД (ГОСТ 19.101 – 77 [1], ГОСТ 19.103 – 77 [2], ГОСТ 19.104 – 78* [3], ГОСТ 19.105 – 78* [4], ГОСТ 19.504 – 79* [5]).

[1] ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов
[2] ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов
[3] ГОСТ 19.104-78* ЕСПД. Основные надписи
[4] ГОСТ 19.105-78* ЕСПД. Общие требования к программным документам
[5] ГОСТ 19.504-79* ЕСПД. Руководство программиста. Требования к содержанию и оформлению

# СОДЕРЖАНИЕ

# 1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

## 1.1. Назначение программы

Программное обеспечение предназначено для виртуального моделирования поведения разных конфигураций систем: ПУ, ПБУ, МФР и целей в воздушной обстановке. Обеспечивает настройку конфигурации по следующим параметрам:

1) настройка координат объектов симуляции;
2) настройки траекторий объектов целей;
3) настройка максимального времени моделирования.

## 1.2. Функции, выполняемые программой

1) моделирование последовательных по времени состояний системы;
2) логирование процесса моделирования в консоль;
3) обмен данными через сообщения объектов симуляции через диспетчер в процессе моделирования;
4) отображение результатов моделирования в виде траекторий подвижных объектов симуляции;
5) сохранение кофигурации системы истории сообщений симуляции в файловую БД;
6) реализация алгоритмов следования целей траекториям;
7) реализация алгоритмов пусков по целям объектами ПУ;
8) реализация алгоритмов управления ПБУ.

## 1.3. Условия необходимые для выполнения программы

### 1.3.1. Объём оперативной памяти

Рекомендуемый объём оперативной памяти 2 Гб или выше. Свободного места на диске необходимо 512 Мб или выше.

### 1.3.2. Требование к программному обеспечению

Специальные программные средства, используемые СПО «Шуршунчик» должны быть представлены локализированной версией операционной системы Windows 10 или Windows 11.

### 1.3.3. Требования к аппаратному средству выполнения

Для исполнения СПО «Шуршунчик» видеокарта не требуется. Процессор выполнения Core 2 Duo 2.2 ГГц или двухядерный AMD 2.5 ГГц.

# 2. ХАРАКТЕРИСТИКА ПРОГРАММЫ

## 2.1. Описание основных характеристик программы

### 2.1.1 Режим работы программы

Специальной программное обеспечение моделирования боевой воздушной обстановки и реагирования зенитной ракетной системы «Шуршунчик» обеспечивает работу в единственном режиме функционирования, который является штатный режим.

В штатном режиме должна обеспечиваться доступность всех функций СПО «Шуршунчик». В штатном режиме функционирования СПО обеспечивает работу пользователей в непрерывном.

### 2.1.2. Контроль правильности выполнения программы

Контроль правильности выполнения программы осуществляется визуально. В случае корректной работы программы после её запуска на экране отображается пустое поле конфигурации со списком возможных моделей средств. После входа в главное окно системы необходимо проверить работоспособность программы (кнопки меню, выполнение операций и т.д.). В работоспособном состоянии программы на экране отображается главное окно приложения без отображения пользователю сообщений о сбое в работе.

### 2.1.3. Средства контроля самовосстанавливаемости программы

Самовосстанавливаемость программы обеспечивается средствами перезапуска или предварительным сохранением входных данных. Перезапуск программы осуществляется через остановку и последующий запуск. Контроль восстановления программы возможно проводить по работоспособности приложения.

## 2.2 Описание основных особенностей программы

Специальное программное обеспечение «Шуршунчик» имеет пользовательский интерфейс, однако не предоставляет конечному пользователю возможности настройки и изменения своих параметров, не относящихся к процессу моделирования.

# 3. ОБРАЩЕНИЕ К ПРОГРАММЕ И ВЫПОЛНЕНИЕ

Собранная программа состоит из архива, в котором должны находится следующие файлы:

1) ./logs/
2) ./SimulationAppUI/images/
3) main.exe

В папке logs сохраняется процесс логирование действий внутри программы после её запуска.

В SimulationAppUI/images/ лежат изображения иконок, используемых в для отображения программы.

Файл запуска программы main.exe не требует каких-либо других файлов, кроме как упомянутых выше.

Обращение к программе после запуска происходит со стороны оператора. Соответствующее описание есть в Руководстве оператора к данному программному обеспечению.

# 4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

## 4.1. Организация используемой входной информации

Входная информация вводится в соответствующих окнах ввода данных или окне «Отображения конфигурации» графического приложения.

## 4.2. Справочная информация по входным данным модулей

В таблице 1 отражены входные параметры модулей программного кода.

Таблица 1. Входные параметры модулей программного кода

| Модуль | Параметр | Описание |
|---|---|---|
| AeroEnv | dispatcher | ссылка на объект диспетчера |
| | ID | собственный ID |
| | targets | список целей класса Movable |
| StartingDevice | dispatcher | ссылка на объект диспетчера |

| | | |
|---|---|---|
| | ID | собственный ID |
| | pos | координата ПУ (x,y,z) |
| | aero_env | ссылка на объект ВО |
| RadarRound | dispatcher | ссылка на объект диспетчера |
| | ID | собственный ID |
| | cp_ID | ID ПБУ |
| | aero_env | ссылка на объект ВО |
| | pos | координаты положения РЛС, относительно глобальной СК (x, y, z) |
| | pan_start | начальное положение по углу поворота относительно глобальной СК в градусах |
| | tilt_start | начальное положение по углу наклона относительно глобальной СК в градусах |
| | view_distance | дальность обзора РЛС |
| | pan_per_sec | угол раскрыва по углу поворота за секунду обзора в градусах |
| | tilt_per_sec | угол раскрыва по углу наклона за секунду обзора в градусах |
| RadarSector | все, что в RadarRound | |
| | pan_angle | максимальный угол раскрыва по азимуту |
| | tilt_angle | максимальный угол раскрыва по углу наклона |
| | type_of_view | horizontal – горизонтальный тип обзора для секторного РЛС (рисунок 1, а), vertical - вертикальный тип обзора для секторного РЛС (рисунок 1, б). |
| CombatControlPoint | dispatcher | ссылка на объект диспетчера |
| | ID | собственный ID |
| | starting_devices_coords | координаты всех ПУ |
| GuidedMissile | dispatcher | ссылка на объект диспетчера |
| | ID | собственный ID |

| | | |
|---|---|---|
| | pos | координата ракеты (x,y,z) |
| | aero_env | ссылка на объект ВО |
| | speed | скорость ракеты |
| | life_time | возможное время жизни ракеты |
| | expl_radius | радиус взрыва ракеты |
| | size | характерный размер ракеты |

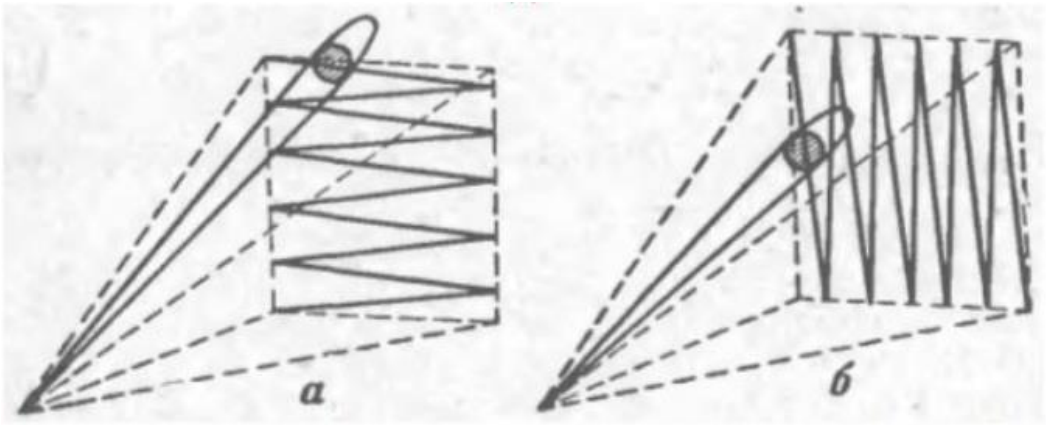


Рисунок 1 – Траектория движения ДНА при зигзагообразном обзоре: скорость переносного движения больше скорости относительного движения (а) и скорость переносного движения меньше скорости относительного движения (б)

## 4.3. Организация используемой выходной информации

Выходная информация поступает в графический интерфейс приложения в окно «Отображение траекторий».

## 4.4. Справочная информация по выходным данным модулей

В таблице 2 отражены выходные параметры модулей программного кода.

Таблица 2. Выходные параметры модулей программного кода

| **Выходные данные** | | | |
|---|---|---|---|
| модуль | параметр | описание | способ получения |
| AeroEnv | entities | объекты в воздушной обстановке (ЗУР, цели) | AeroEnv.getEntities() |
| RadarRound | visible_objects | объекты, которые | RadarRound.findObjects() |

| | | | |
|---|---|---|---|
| | | видит Радар | |
| RadarSector | visible_objects | объекты, которые видит Радар | RadarSector.findObjects() |
| CombatControlPoint | obj_type, sim_obj_key | находит наиболее похожий объект из памяти иначе считается, что новый объект это новая цель | CombatControlPoint.findMostSimilarObject() |
| GuidedMissile | status | Узнать статус ракеты: 0 - не активна, 1 – летит, 2 - поразила цель, 3 - пропустила цель, закончилось топливо | GuidedMissile.getStatus() |

# 5. СООБЩЕНИЯ ПРОГРАММИСТУ

## 5.1. Формат получения сообщений

Программист получает сообщения в формате логов в папке logs.

## 5.2. Справочная информация по сообщениям программисту

В таблицах 3.1 – 3.7 отражены все форматы сообщений, поступающих в процессе логирования.

Таблица 3.1. Формат логов, выводящихся в процессе моделирования для модуля МФР

| Сообщение | Текст | Как получить |
|---|---|---|
| МФР | | |
| Угол поворота обзора МФР | Radar с id {self._ID} видит по углу поворота от {self.pan_cur} до {self.pan_cur + self.pan_per_sec} | 1. МФР вызывает функцию findObjects()<br>2. МФР вызывает функцию moveToNextSector() |
| Угол наклона обзора МФР | Radar с id {self._ID} видит по углу наклона от {self.tilt_cur} до {self.tilt_cur + self.tilt_per_sec}") | |
| МФР обнаружила объект | Radar с id {self._ID} видит объект с сферическими координатами (dist, pan, tilt): {r, pan, tilt} | МФР вызывает функцию findObjects()<br>Среди объектов ВО находятся те, что лежат в рамках обзора МФР |
| Количество объектов, видимых МФР | Radar с id {self._ID} видит {len(visible_objects)} объектов | МФР вызвала функцию findObjects() и обработала все объекты в ВО |
| Количество полученных МФР сообщений от ПБУ | Radar с id {self._ID} получил {len(msgs_from_ccp)} сообщений от CombatControlPoint | МФР вызывает функцию changeMissileCoords() и проверяет наличие сообщений с id = MSG_CCP2RADAR_type |
| Количество полученных МФР сообщений от ЗУР | Radar с id {self._ID} получил {len(msgs_from_gm)} сообщений от ЗУР о том что она перестала существовать | МФР вызывает функцию sendCcpHitMissile() и проверяет наличие сообщений с id = MSG_GM2RADAR_type |
| МФР отправило сообщение ПБУ с целями | Radar с id {self._ID} отправил сообщение CombatControlPoint с id {self.cp_ID} с видимыми объектами | МФР находит все цели и отправляет сообщение Radar2CombatControlMsg со списком целей |
| МФР отправляет рисовальщику список с положениями видимых целей | Radar с id {self._ID} отправил сообщение Drawer с id {TARGET_TYPE_DRAWER} с положениями видимых объектов | МФР отправила сообщение Radar2DrawerMsg рисовальщику с положением всех найденных целей |

| | | |
|---|---|---|
| МФР отправила диспетчеру список с положениями видимых целей | Radar с id {self._ID} отправил сообщение Dispatcher с id {DISPATCHER_ID} с положениями видимых объектов | МФР отправила сообщение Radar_ViewMessage диспетчеру с положением всех найденных целей |

Таблица 3.2. Формат логов, выводящихся в процессе моделирования для модуля ВО

| ВО | | |
|---|---|---|
| Количество объектов | AeroEnv имеет {len(self.entities)} | AeroEnv вызывает runSimulationStep() |
| ВО отправила цели и зур ГУИ | ВО отправил GUI: {len(missile_list)} ЗУР и {len(target_list)} ЦЕЛЕЙ | ВО вызывает функцию send_vis_objects2gui() и отправляет сообщение AeroEnv_ViewMessage |
| Количество объектов до взрыва | AeroEnv количество объектов {len(self.entities)} | ВО вызывает функцию explosion() |
| Состояние объекта до взрыва | AeroEnv расстояние между ЗУР и объектом {dist(pos, el.pos)}, размер объекта {el.size}, радиус взрыва {expl_rad} | ВО вызывает функцию explosion()<br>и для каждого объекта сообщает расстояние до зур, размер объекта и радиус взрыва |
| Взрыв цели | AeroEnv взрыв ЗУР с координатами: {pos}, Уничтожила цель с ID: {el._ID} и координатами {el.pos} | В ходе функции explosion()<br>после нахождения всех подходящих целей они уничтожаются |
| Самолету назначена новая контрольная точка | Самолет ID: {self._ID}, Новая контрольная точка: {self.target_point} | Если у самолёта остались контрольные точки - ему назначается новая |
| Самолет прошел все контрольные точки | Самолет ID: {self._ID} прошел все контрольные точки, летит по вектору: {self.vel} | Если у самолета не осталось контрольных точек - сообщается по какому вектору он летит |
| | Самолет ID: {self._ID}, координаты Самолета: {self.pos} корректировка большого угла поворота | |

| | | |
|---|---|---|
| | {round(np.rad2deg(angle_between(self.vel, vel_old)), 2)}° | |

Таблица 3.3. Формат логов, выводящихся в процессе моделирования для модуля ПБУ

| ПБУ | | |
|---|---|---|
| Координаты ПБУ | starting_devices_coords {starting_devices_coords} | Инициализация ПБУ |
| Создана ПБУ | Создан ПБУ с ID {ID} | |
| Цели, видимые ПБУ | ПБУ видит объект, {max(0, target_speed_mod *(time_went - tick)- obj_error)}, {coord_dif}, {max(0, target_speed_mod * (time_went + tick)+ obj_error)} | ПБУ вызывает findMostSimilarObject() и сообщает о видимых целях |
| Ракеты, видимые ПБУ | ПБУ видит объект, {max(0, missile_speed_mod *(time_went - tick)- obj_error)}, {coord_dif}, {max(0, missile_speed_mod * (time_went + tick)+ obj_error)}" | ПБУ вызывает findMostSimilarObject() и сообщает о видимых ракетах |
| Типы объектов видимых ПБУ | ПБУ решила что объект с координатами {obj_coord} это {obj_type}, ЗУР - 2, Цель старая - 1, Цель новая - 0 | ПБУ показывает классификацию объектов в конце функции findMostSimilarObject() |
| ПБУ спросилла сколько ЗУР у ПУ | ПБУ спросила ПУ с id {key} сколько у нее ЗУР | ПБУ отправляет ПУ сообщение msg2starting_device в request_starting_devices_capacities() |
| Количество сообщений от ПУ | ПБУ получил от ПУ {len(missile_capacity_msg)} сообщений о кол-ве ЗУР | В request_starting_devices_capacities() ПБУ проверяет количество сообщений с id= MSG_CCP_MISSILE_CAPACITY_ type |
| ПБУ получил сообщение о количестве ЗУР от ПУ | ПБУ получил от ПУ {msg.sender_ID} сообщений о {msg.missile_number} ЗУР | В get_starting_devices_capacity() ПБУ проверяет наличие сообщений с id= MSG_CCP_MISSILE_CAPACITY_ type |

| | | |
|---|---|---|
| Количество сообщений от Радара об уничтожени и ЗУР | ПБУ получил {len(msgs_missile_hit)} сообщений от Радара о том что ЗУР перестала существовать | В get_hit_guided_missiles() ПБУ проверяет наличие сообщений с id= MSG_RADAR2CCP_GM_HIT_type |
| Количество сообщений от ПУ о запуске ЗУР | ПБУ получил от ПУ {len(msgsFromStartingDevice)} сообщений о запуске ЗУР | В get_launched_missiles() ПБУ проверяет наличие сообщений с id= MSG_SD2CCP_MS_type |
| | {target_num_dict}, {len(self.target_dict.items())} | |
| ПБУ получил id и начальные координаты ЗУР от ПУ | ПБУ получил от ПУ id ЗУР:{missile_id}, начальные координаты ЗУР:{missile_coord} | В get_launched_missiles() ПБУ проверяет наличие сообщений с id= MSG_SD2CCP_MS_type и обрабатывает каждое из них |
| ПБУ отправил сообщение GUI со списком целей и ЗУР | ПБУ отправил {len(target_list) + len(missile_list)} смс GUI | В send_vis_objects2gui() ПБУ отправляет GUI сообщения CombatControlPoint_ViewMessage |
| ПБУ отправил сообщение Рисовальщи ку с координатам и целей и ЗУР | ПБУ отправил {len(list_for_drawer)} смс Рисовальщику | В send_vis_objects2drawer() ПБУ отправляет GUI сообщения CombatControl2DrawerMsg со списком [тип, координаты] |
| ПБУ получил сообщения от МФР | ПБУ получил сообщения от {len(msg_from_radar)} МФР | ПБУ получил сообщения от МФР с id= MSG_RADAR2CCP_type и отправляет их количество |
| | ПБУ получил {len(msg.visible_objects)} сообщений от МФР с id {msg.sender_ID} | ПБУ отправляет сообщение, обрабатывая каждое от МФР |
| ПБУ отправил ПУ координаты новой цели | ПБУ отправил ПУ id {sd_id} координаты новой цели: {obj_coord} | ПБУ отправляет ПУ сообщение CombatControl2StartingDeviceMsg с координатами новой цели |

| | | |
|---|---|---|
| Не осталось ЗУР для целей | Не осталось свободных ЗУР!!! | Есть новая цель, а ЗУР нет! |
| ПБУ сообщает МФР, что у ЗУР есть обновленные координаты цели | ПБУ отправил сообщение Радару, что у ЗУР с id:{missile_id}, новые координаты ее цели:{obj_coord} | ПБУ отправляем МФР сообщение CombatControl2RadarMsg с ЗУР и её новыми координатами |
| ПБУ передает новые координаты старой ЗУР | ПБУ увидел старую ЗУР с id:{self.missile_dict[sim_obj_key].id}, новые координаты:{obj_coord} | ПБУ передает новые координаты старой ЗУР |

Таблица 3.4. Формат логов, выводящихся в процессе моделирования для модуля рисовальщика

| Рисовальщик | | |
|---|---|---|
| Рисовальщик получает сообщение от ПБУ | Рисовальщик получил {len(CCP_list)} сообщений от ПБУ | Рисовальщик получает сообщение от ПБУ с id= MSG_CCP2DRAWER_type |

Таблица 3.5. Формат логов, выводящихся в процессе моделирования для модуля ЗУР

| ЗУР | | |
|---|---|---|
| Запуск ЗУР | Запуск. ЗУР ID: {self._ID}, начальная позиция: {self.pos}, начальная позиция цели: {self.pos_target} | Вызов функции launch() |
| Корректировка большого угла поворота | ЗУР ID: {self._ID}, координаты ЗУР: {self.pos}, корректировка большого угла поворота {round(np.rad2deg(angle_between(self.vel**,** vel_old))**,2**)}° | Вызов функции updateCoordinate(), если угол между векторами скоростей > GuidedMissile_MaxRotAngle |
| Информация о ЗУР | ЗУР ID: {self._ID}, pos_target {self.pos_target}, target_vel {self.target_vel}, pos {self.pos} | Вызов функции checkIsHit() |
| Положение ЗУР | ЗУР ID: {self._ID}, координаты ЗУР: {self.pos}, расстояние до цели: {(((self.pos - self.pos_target) ** | |

| | | |
|---|---|---|
| относительно цели | 2).sum()) ** 0.5}, расстояние до цели интерполированное {(((self.pos_target+self.target_vel*self._simulating_tick*self.delay_time - self.pos) ** 2).sum())**0.5} | |
| Обработка сообщений от МФР | ЗУР ID: {self._ID}, координаты ЗУР: {self.pos}, получила сообщение от Радара ID: {self.radar_id}, новые координаты цели: {pos_target}, ее вектор скорости: {target_vel} | Вызов функции runSimulationStep() Имеются сообщения от МФР |
| ЗУР поразила цель | ЗУР ID: {self._ID}, координаты ЗУР: {self.pos}, поразила цель с координатами: {self.pos_target} | Вызов функции runSimulationStep() Статус ЗУР = 1 сменился на статус = 2 после вызова функции checkIsHit() |
| ЗУР сообщила ПБУ об уничтожении цели | ЗУР ID: {self._ID}, отправила сообщение Радару {self.radar_id} о том, что она перестала существовать | |
| У ЗУР закончилось топливо (время жизни истекло) | ЗУР ID: {self._ID}, координаты ЗУР: {self.pos}, пропустила цель с координатами: {self.pos_target. Кончилось топливо | Вызов функции runSimulationStep() Статус ЗУР = 1 не сменился после вызова функции checkIsHit() и время существования превысило допустимое время жизни |
| ЗУР уничтожила цель | ЗУР ID: {self._ID} прекратила существоавние из-за поражения цели | Вызов функции runSimulationStep() Статус ЗУР = 2 после отработки всех условий и функций |
| ЗУР не хватило топлива | ЗУР ID: {self._ID} прекратила существоавние из-за поражения цели нехватки топлива | Вызов функции runSimulationStep() Статус ЗУР = 3 после отработки всех условий и функций |

Таблица 3.6. Формат логов, выводящихся в процессе моделирования для модуля Диспетчера

| Диспетчер | | |
|---|---|---|
| Номер текущего шага | Номер текущего шага: {self.__current_step} | Вызов функции run(), номер данной итерации |

Таблица 3.7. Формат логов, выводящихся в процессе моделирования для модуля ПУ

| ПУ | | |
|---|---|---|
| Получены сообщения с вопросом о количестве ЗУР | ПУ с ID {self._ID} получила вопрос о кол-ве ЗУР | Вызов функции answer2request_gm_capacity()<br>Есть сообщения типа = MSG_CCP_MISSILE_CAPACITY_type<br>ПУ отправляет сообщения типа MissileCapacityMsg |
| ПУ отправляет количество оставшихся ЗУР ПБУ | ПУ с ID {self._ID} отправила ПБУ сколько у ПУ ЗУР | |
| Количество сообщений от ПБУ | ПУ с ID {self._ID} получила {len(new_messages)} сообщений от ПБУ | Вызов функции runSimulationStep()<br>Проверка наличия сообщений с типом= MSG_CCP2SD_type<br>Запуск ЗУР при их наличии по переданным координатам |
| Запуск ЗУР | ПУ с ID {self._ID} запустила зур с ID {free_missiles[0]._ID} и отправила сообщения ПБУ с ID {msg.sender_ID} | |

# ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

**МФР** – многофункциональный радиолокатор.
**ПБУ** – пункт боевого управления.
**ПУ** – пусковая установка.
**ЗУР** – зенитная управляемая ракета.
**СПО** – специальное программное обеспечение.
**ДНА** – диаграмма направленности антенны.
**ВО** – воздушная обстановка.

# ДИАГРАММА КЛАССОВ

В рисунках 2.1, 2.3 представлена диаграмма классов, описывающая весь программный модуль СПО «Шуршунчик».

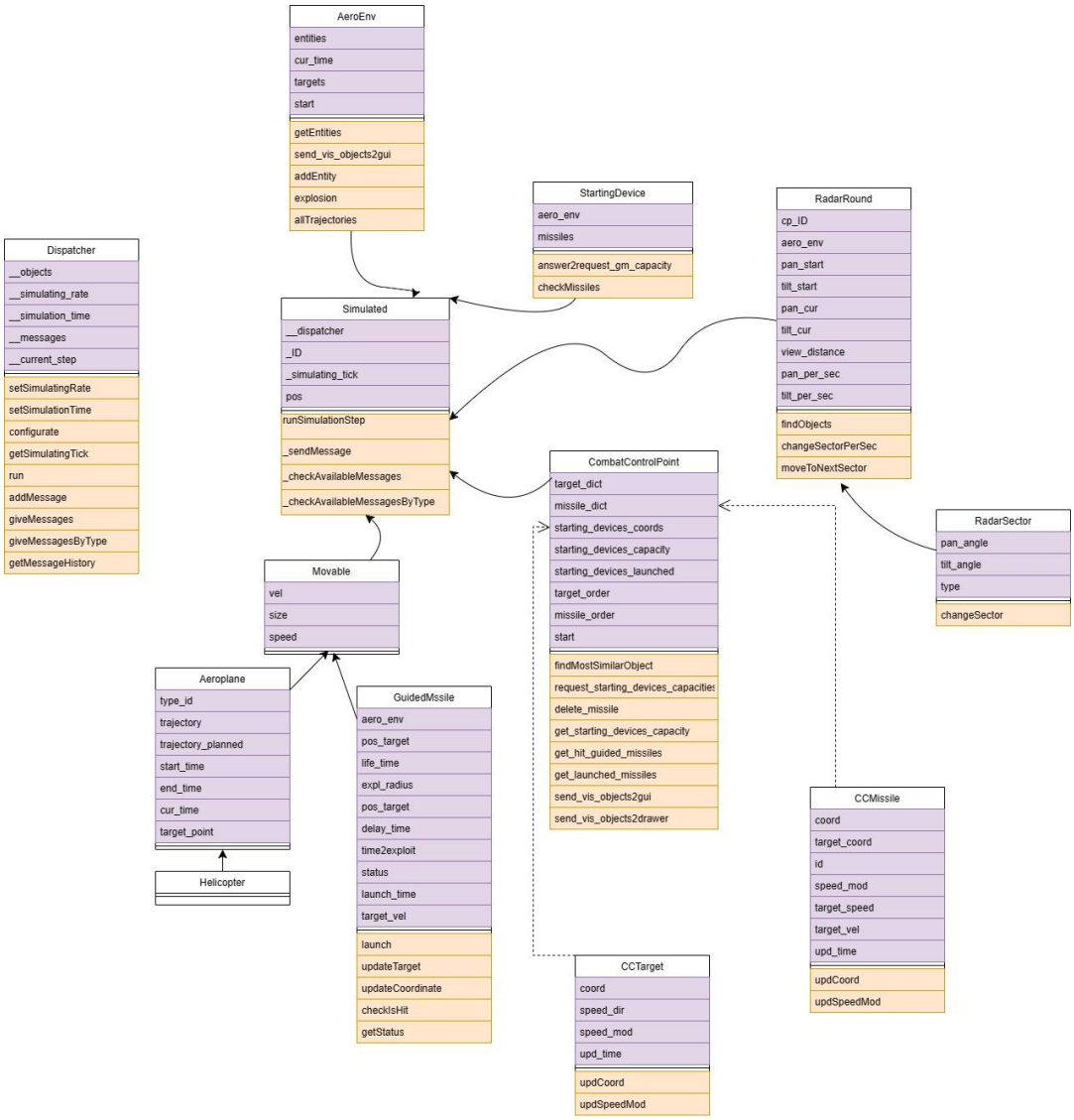


Рисунок 2.1 – Диаграмма классов модулей

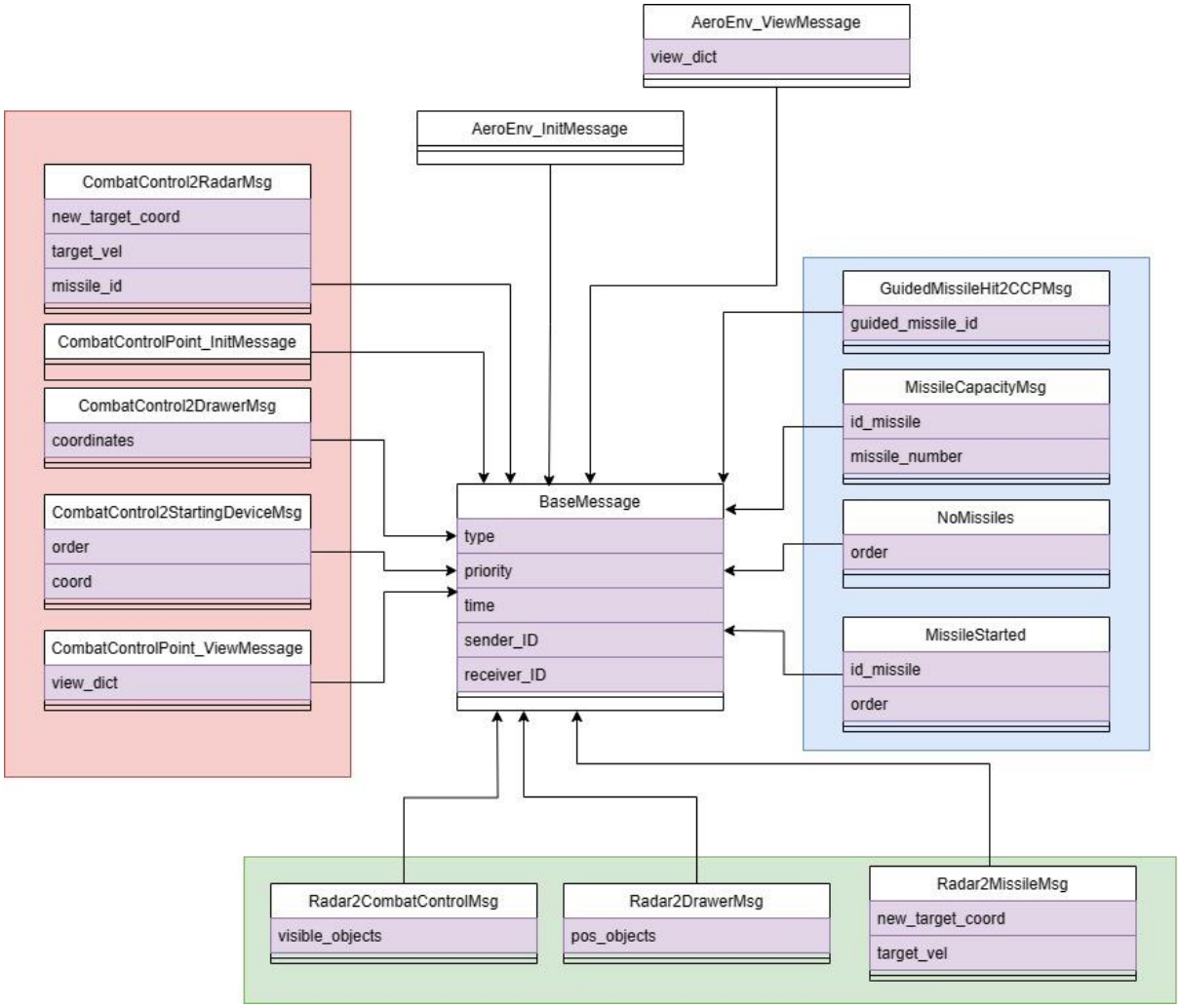


Рисунок 2.2 – Диаграмма классов сообщений