**Класс Missile:**

(Всё в паблик - секции)

* Текущее положение ракеты
* Скорость ракеты
* Направление движения (трёхмерный вектор)
* Положение цели
* ID цели
* История положений ракеты
* Булева переменная взорвана / не взорвана

**Функции класса Missile:**

* Конструктор (положение ракеты, положение цели, ID цели, скорость ракеты)
* Деструктор ()
* Обновление ()

**Реализация функций класса Missile:**

* Конструктор (положение ракеты, положение цели, ID цели, скорость ракеты):

{

Заполнение соответствующих полей;

По умолчанию ракета не взорвана;

Формируется вектор направления полёта ракеты;

Дополняем вектор положений ракеты.

}

* Обновление ():

{

Если положение ракеты отстоит от положения цели на малую величину, тогда ракета взрывается;

Направление движения есть положение цели – текущая позиция (+ нормализуем);

Обновление текущей позиции: она равна направление \* скорость \* ступеньку времени симуляции;

Если ракета перелетела цель, тогда она взрывается, её позиция становится равной позиции цели (?)

История положения ракеты дополняется текущей позицией.

}

**Класс Launcher:**

(Всё в паблик - секции)

* Положение Пусковой установки (ПУ)
* Её имя (т.к. возможно наличие нескольких ПУ)
* Таймер (установка имеет время перезарядки)
* Статус пусковой установки (занята / свободна)
* Количество ракет - число
* Вектор из структур (с порядком работы по целям (созданный ПБУ для данной ПУ):

ID ракеты

Положение цели

ID цели

* Вектор из объектов типа Missile - запущенные ракеты

**Функции класса Launcher:**

* Конструктор (положение ПУ, имя ПУ, Количество ракет)
* Деструктор ()
* Обновление ()
* Запуск ракеты (положение цели, ID цели)

**Реализация функций класса Launcher:**

* Конструктор (положение ПУ, имя ПУ, Количество ракет):

{

Заполняет соответствующие поля;

Устанавливает Собственное время = 0;

Устанавливает Статус – ПУ свободна.

}

* Обновление ():

{

Очищается вектор (из Missile) запущенных ракет;

Вводится условие на время перезарядки. Если не прошло определённое время с последнего, пуска считать, что ПУ занята и ракету не запускать (при этом статус - занята);

Если в очереди есть цели на обработку и пусковая установка не занята, тогда запускаем ракету по цели. Удаляем объект из очереди. Выставляем таймер. Статус ПУ – занята.

}

* Запуск ракеты (положение цели, ID цели):

{

Если количество ракет ненулевое, тогда уменьшаем на 1 количество ракет и запускаем ракету;

При этом ей передаётся положение ПУ, положение цели, ID цели, скорость ракеты;

Ракета добавляется в список запущенных ракет.

}

**Интерфейс работы Ракеты с Диспетчером модели:**

Диспетчер вызывает функцию update() применительно ко всем ракетам (объектам класса «Missile») на каждом такте обновления модели.

При этом при обращении к полю «log\_position» объекта класса «Missile» получаем вектор историй положения ракеты. (=> можем отрисовать движение ракеты)

**Интерфейс работы ПУ с Диспетчером модели:**

Диспетчер создаёт объекты класса «Launcher». При этом в конструктор передаются:

1. положение ПУ;
2. имя, присвоенное ПУ;
3. количество ракет в данной ПУ.

Данные о созданной ПУ передаются в ПБУ Диспетчером.

Затем Диспетчер вызывает функцию update() применительно ко всем ПУ (объектам класса «Launcher») на каждом такте обновления модели.

Диспетчер получает данные о выпущенных ракетах путём обращения к полю «missiles» (список из объектов класса «Missile») объекта класса «Launcher».

ВАЖНО! При вызове «Объект Launcher».update() очищается вектор из запущенных ракет. Его нужно сохранить в диспетчере в каком-то виде.

**Интерфейс работы ПУ с ПБУ:**

ПБУ создаёт список из объектов класса «Launch\_event». При этом в каждый объект передаются:

1. ID ракеты;
2. положение цели (когда было принято решение об её обстреле);
3. ID цели.

Объект[0] обрабатывается первым.

ПБУ передаёт созданный список объекту класса «Launcher» путём обращения к полю «order» (т.е. это тоже список из объектов класса «Launch\_event»).

Дополнения:

Наладить взаимодействие с локатором! (для ракеты при отслеживании цели)