Руководство по созданию автоматических сценариев в MAP21

Оглавление

Введение

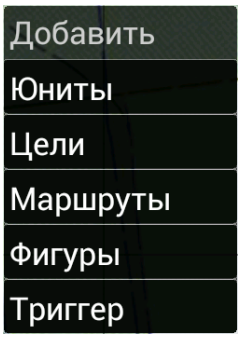
Для автоматизации игрового процесса в системе MAP21 предусмотрены триггеры и скриптовый язык.

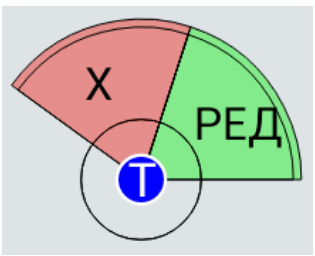
Триггеры – это специальные объекты на карте, которые видны только пользователям с правами «Администратор» и используются для выполнения цепочки команд (так называемого «**скрипта**») при наступлении определенного условия. Условиями могут быть расстояние или азимут между метками на карте, их статус, видимость, вхождение в область определенной фигуры и т.п. В качестве команд может быть смена статуса, видимости, положения меток, отправка сообщений и приказов пользователям и т.п.

На практике триггеры могут быть использованы для автоматизации таких процессов как: условные «минные» поля, предупреждение о выходе за игровую территорию или о подходе противника, автоматическое возрождение в определенной зоне с указанной задержкой по времени, отображение плана следующей миссии при выполнении целей предыдущей, анализ захвата территорий и подсчет времени или количества этих захватов, а также многое другое. Специально разработанный набор команд имеет большой потенциал и позволит организаторам игр эффективно использовать функционал системы.

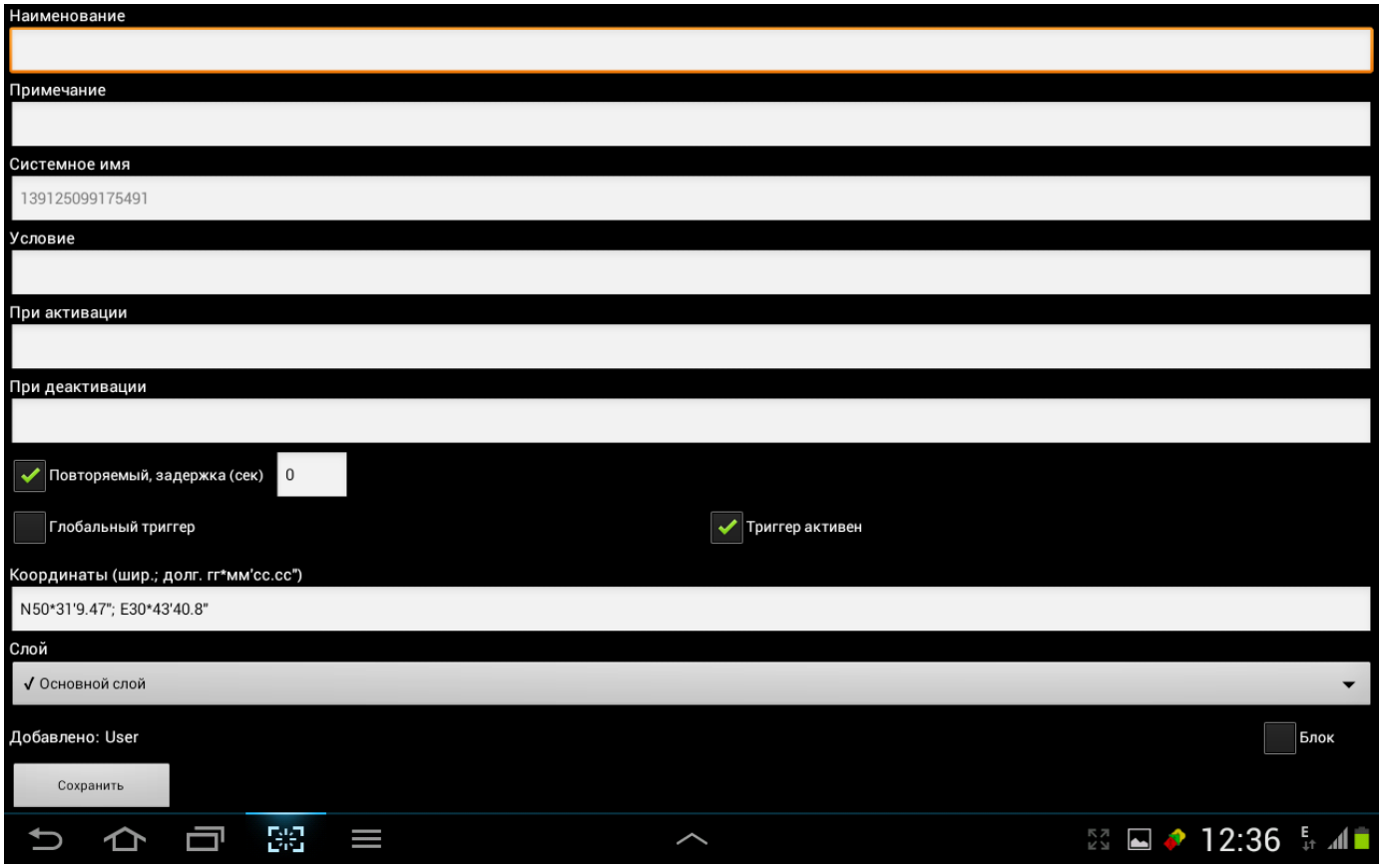
Добавление триггера

Чтобы добавить триггер необходимо зажать палец на выбранном пустом месте карты и в появившемся меню «Добавить» выбрать пункт «Триггер». В результате на карте появится метка триггера и его секторное меню.





Нажмите опцию «РЕД», чтобы зайти в редактирование атрибутов триггера.



У каждого триггера в атрибутах есть:

- **Наименование** - для визуального различия триггеров на карте;
- **Примечание** - для подробного описания назначения триггера;
- **Системное имя** - для обращения к атрибутам триггера из скриптов;
- **Условие** активации триггера, состоящее из конструкций, описанных в данном руководстве ниже;
- Скрипт **при активации**, который срабатывает, если вышеуказанное условие становится истинно;
- Скрипт **при деактивации**, который срабатывает, если вышеуказанное условие перестает быть истинно, то есть становится ложно.

Триггер может быть в одном из двух состояний - активирован или деактивирован. Активация наступает в момент, когда условие становится истинным. При этом происходит переключение состояния триггера и вызов скрипта по активации. Пока условие не станет ложно, скрипт активации больше не выполняется. Если условие становится ложно, то, соответственно, выполняется скрипт по деактивации.

Триггеры бывают **одноразовые** и **повторяемые**. У повторяемого триггера скрипт при активации и деактивации выполняется неограниченное число раз, а у одноразового – только один. Повторяемому триггеру можно задать паузу между срабатываниями. Режим **повторяемости и задержка** в секундах указывается в соответствующих атрибутах триггера.

Еще триггеры делятся на **локальные** и **глобальные**. Локальные триггеры срабатывают на каждом устройстве пользователя в отдельности, а глобальные срабатывают только на том устройстве, где условие сработало первым по времени, после чего факт срабатывания фиксируется на сервере и на остальных устройствах данный триггер игнорируется. Эта функция переключается опцией «**Глобальный триггер**» на экране свойств.

Так же, триггер может быть **включен** или **выключен**. У выключенного триггера не проверяется его условие активации и, соответственно, не срабатывает ни скрипт при активации, ни скрипт при деактивации. Статус включенности триггера определяется флажком «**Триггер вкл.**», а так же может быть изменен программно из скрипта (см. ниже).

Принцип построения скрипта

Для составления условий триггеров можно использовать операторы и функции, описанные в следующем разделе данного руководства.

В качестве скрипта можно использовать набор команд и функций, описанных в следующем разделе данного руководства. Команды можно писать как в одну строчку, так и с переносом. Команды разделяются символом ";". Для последней команды скрипта или блока разделитель можно не ставить. Блоки команд выделяются фигурными скобками “{ }”. После закрытия блока команд разделитель не ставится.

If (checkStatus(@goal1,CMP)) { hide(@mission1); show(@mission2); sayTeam(Player,”Mission1 complete!”) }

Триггер имеет кэш переменных, которые позволяют сохранять в него цифровые целочисленные значения, произвольные строки, идентификаторы объектов или их множества. С помощью переменных кэша можно отмерять прошедшее время или считать количество активаций триггера, хранить промежуточные значения координат, списки меток и т.п. Все переменные в кэше имеют строковый тип: числовые значения хранятся как строки; имена объектов хранятся как строки, начинающиеся на символ «@»; координаты хранятся как текст во внешнем формате, указанном в настройках программы, например, «N00*00’00.0”; E00*00’00.0”»; множества как перечень наименований объектов в квадратных скобках через запятую (см. ниже). Для локальных триггеров кэш очищается при перезапуске программы, поэтому лучше использовать кэш глобальных триггеров.

(Пример использования кэша…)

В програмне доступны множества объектов, задаваемые следующим образом «[<@объект1>,<@объект2>,…<@объектN>]». Множества используются для выполнения серии одинаковых команд над несколькими объектами. Кроме изначального ручного задания множества, его можно получить с помощью функции getUnits. Множеством можно управлять поэлементно с помощью команд Add и Remove.

Оператор **forEach**(<@множество>) {<блок>} позволяет выполнить определенный набор команд для каждого объекта в множестве. При этом переменная @Object содержит текущий элемент множества.

Если скрипт содержит ошибку, пользователь увидит сообщение, в котором будет описано наименование скрипта, строка с ошибкой и другие детали.

Список скриптовых конструкций

Логические операторы

При написании условий можно использовать следующие логические операторы:

- **AND** - логическое «И»;
- **OR** - логическое «ИЛИ»
- **=** - равно;
- **!=** - неравно;
- **>=** - больше или равно;
- **<=** - меньше или равно;
- - больше;
- **<** - меньше;

В качестве логических значений используются числа по принципу – все, что больше 0, истинно, а остальное - ложно.

Арифметические операции

Для вычисления различных формул можно использовать следующие арифметические операции:

- **«+»** - сложение
- **«-»** - вычитание
- **«*»** - умножение
- **«/»** - целочисленное деление

В случае если блоки операндов отделены скобками "( )", тогда выполнение действий начинается с самых внутренних скобок. По умолчанию действия производятся слева направо.

Например: 2+4*2 даст в результате 12 - сначала выполнится сложение, затем умножение. Чтобы посчитать классически, нужно написать 2+(4*2). Если в качестве аргументов логических выражений используются вычисляемые значения, то лучше их брать в скобки, например, (1+3) OR (2-4).

Операторы условного перехода

Для ветвления хода выполнения скрипта можно использовать следующую конструкцию условного перехода:

if (<условие>) **then** {<блок1>} **else** {<блок2>};

Если условие истинно, тогда выполняется <блок1>, иначе выполняется <блок2>.

Можно использовать сокращенную версию:

if (<условие>) **then** {<блок1>}

Вместо блока можно писать одну команду без фигурных скобок.

Константы

В процессе написания скрипта вы будете оперировать некоторыми понятиями, которые определены фиксированными константами.

Глобальные объекты:

- **@Player** – объект, обозначающий метку текущего пользователя, на устройстве которого отрабатывает скрипт.
- **@Trigger** – объект, указывающий на триггер, в котором выполняется текущий скрипт.
- **@Null –** пустой не существующий объект. Для проверки наличия лидера, объекта навигации и т.п.
- **@Object** – используется как указатель на текущий элемент внутри блока оператора **forEach.**

Статусы целей:

- **ACT** - (actual) актуально;
- **CMP** - (completed) выполнено;
- **FLD** - (failed) провалено.

Статусы игроков:

- **REA** - (ready) готов;
- **CLR** - (clear) чисто;
- **DNG** - (danger) в опасности;
- **DMG** - (damage) ранен;
- **DEA** - (dead) погиб.

Статусы триггеров:

- **ON** - (on) включен;
- **OFF** - (off) выключен.

Названия возможных операций с интерфейсом, вызываемых командой «shell»:

- **ST_CLEAR** - статус «чисто»;
- **ST_UNDERFIRE** - статус «в опасности»;
- **ST_DAMAGE** - статус «ранен»;
- **ST_DEAD** - статус «убит»;
- **ST_READY** - статус «готов»;
- **MSG_1** - сообщение «Готов»;
- **MSG_2** - сообщение «Принял»;
- **MSG_3** - сообщение «Выполняю»;
- **MSG_4** - сообщение «Не могу»;

Действия:

- **FB** (fall back) – в строй;
- **MV** (move) – двигайтесь;
- **AT** (attack) – атакуйте;
- **HL** (hold) – займите;
- **FN** (find) – найдите;
- **ST** (stop) – отбой.

Цвета:

- **BL** – синий;
- **RD** – красный;
- **GR** – зеленый;
- **AZ** – голубой;
- **YW** – желтый;
- **WH** – белый;
- **BK** – черный;

Размерность группы:

- **0** – снаряжение;
- **1** – боец;
- **2** – звено;
- **3** – отделение;
- **4** – секция;
- **5** – взвод;
- **6** – рота;
- **7** – батальон;
- **8** – полк.

Звания пользователей:

- **PVT** – (private) рядовой;
- **CPL**- (corporal) капрал;
- **SGT** – (sergeant) сержант;
- **LT** – (leutenant) лейтенант;
- **CPT** – (captain) капитан;
- **MJR** – (major) майор;

- **COL** – (colonel) полковник.

Специализации групп:

- **IF** – (infantry) пехота;
- **CV** – (cavalry) кавалерия;
- **AR** – (armor) бронетехника;
- **AY** – (artillery) артиллерия;
- **EN** – (engineer) инженеры;
- **AV** – (aviation) авиация;
- **SO** – (spec ops) спецназ;
- **IN** – (intel) разведка;
- **PL** – (police) полиция;
- **AA** – (anti air) ПВО;
- **RD** – (radio) связь.

Специализации бойцов:

- **RM** – (rifleman) стрелок;
- **MG** – (machinegunner) пулеметчик;
- **GR** – (grenadier) гренадер;
- **SR** – (sniper) снайпер;
- **MR** – (mortar) минометчик;
- **AT** – (antitank) ПТ боец;
- **EN** – (engineer) инженер;
- **UN** – (unknown) неизвестный.

Тип объекта:

- **UN_FR** – (friend) юнит союзник;
- **UN_EN** – (enemy) юнит враг;
- **UN_NE** – (neutral) юнит нейтрал;
- **UN_UN** – (unknown) юнит неизвестный;
- **UN_DE** – (dead) юнит мертвый;
- **TG** – (target) цель;
- **WP_ST**– (way point) обычная точка маршрута;
- **WP_AT** – (attack) точка маршрута атака;
- **WP_SP** – (support) точка маршрута поддержка;
- **WP_AM** - (ambush) точка маршрута засада;
- **WP_CO** – (coordination) точка маршрута координация;
- **WP_PI** – (point of interest) точка интереса;
- **FG_LI** – (line point) точка фигуры линии;
- **FG_PO** – (polygon point) точка фигуры полигона;
- **FG_SQ** – (square) квадрат;
- **FG_CI** – (circle) круг.

Функции и команды

Пользователю доступны следующие **функции**:

Not(<значение>) - логическое «НЕ». Если <значение> <= 0, то возвращает «1», в противном случае «0».

getDist(<@объект1>,<@объект2>) – возвращает дистанцию в метрах между двумя объектами.

getAzim(<@объект1>,<@объект2>) – возвращает азимут (число от 0 до 359) от объекта1 на объект2.

getPos(<@объект>) – возвращает строку с координатами объекта в формате, указанном в настройках.

getOffset(<координаты>,<азимут>,<дистанция>,<рандом_азим>,<рандом_дист>) – возвращает координаты со смещением по указанному азимуту на указанное расстояние от текущих. Последние два параметра опциональные и задают верхний порог случайного определения азимута и дистанции. Если один из последних параметров равен «0», то берется точное значение азимута или дистанции, соответственно. Если значение не равно «0», то берется случайно значение между <азимут> и <рандом_азим>, либо <дистанция> и <рандом_дист>.

getDir(<@объект>) – возвращает азимут направления обзора или движения юнита, а так же угол поворота фигуры «квадрат».

getSide(<@объект>) – возвращает цифровой идентификатор стороны объекта. Идентификаторы указаны в настройках программы на вкладке «Сценарий».

getStatus(<@объект>) – возвращает статус юнита, цели или триггера.

getColor(<@объект>) – возвращает цвет фигуры или маршрута.

getSize(<@объект >) – возвращает константу размерности группы (см. константы), либо размер фигуры «квадрат» и «круг».

getType(<@объект>) – возвращает тип любого объекта на карте.

getRank(<@юнит>) – возвращает звание пользователя.

getRole(<@юнит>) – возвращает специализацию юнита.

getTarget(<@юнит>) – возвращает текущий объект навигации у юнита.

getValue(<@имя_триггера.имя_переменной>) - возвращает значение переменной из кэша.

getTeam(<@юнит>) - возвращает идентификатор группы, в которую входит пользователь или группа.

getLeader(<@юнит>) - возвращает идентификатор лидера указанного пользователя или группы.

getUnits(<@юнит>,<вся_ветвь>) – возвращает множество юнитов, входящих в группу или в зону фигуры. Второй параметр может принимать значения «1» или «0» и определяет - возвращать всех игроков или только группы и игроков верхнего уровня, находящихся в зоне, или непосредственно входящих в группу.

Count(<@множество>,<@сторона>) - возвращает количество объектов в множестве. Опциональный параметр – идентификатор стороны. Если сторона указана, то считаются только метки этой строны.

checkStatus(<@объект>,<статус>) - проверяет статусы у целей, игроков и триггеров. Возвращает «1» или «0».

time(<hh:mm:ss>) - возвращает время в миллисекундах. При отправке пустого аргумента возвращает значение текущего времени.

isInside(<@объект>,<@фигура>) – проверяет или объект находится внутри фигуры. Возвращает «1» или «0».

isVisible(<@объект>) – проверяет видимость слоя или объекта на карте. Возвращает «1» или «0».

isGroup(<@юнит>) - проверяет является ли юнит группой или пользователем. Возвращает «1» или «0».

isNull(<@объект>) – проверяет существует ли еще указанный объект. Возвращает «1» или «0».

createObject(<тип>,<координаты>,<название>,<примечание>,<идентификатор>,<слой>,<родитель>) – создает на карте метку определенного типа в указанных координатах, с данным названием и примечанием, на указанном слое. Возвращает идентификатор объекта. Если слой не указан, то метка создается на основном. Если идентификатор не указан, то он присваивается автоматически. Последний параметр – предыдущая точка для создания линий или родительская группа для создания групп. Если тока не указана, то создается новая линия или группа в корневом узле структуры.

Пользователю доступны следующие **команды**:

setPos(<@объект>,<@координаты>) – перемещает объект на карте по указанным координатам.

setColor(<@объект>,<цвет>) – изменяет цвет фигуры или маршрута на указанный.

setSize(<@юнит>,<размер>) – изменяет размерность группы и размер квадрата или круга.

setRank(<@юнит>,<звание>) – устанавливает звание пользователя.

setRole(<@юнит>,<специализация>) – устанавливает специализацию юнита.

setType(<@точка>,<тип>) – устанавливает тип маршрутной точки или юнита. Категории меток не меняет!

setStatus(<@объект>,<статус>) - изменяет статус цели, игрока или триггера на указанный.

setValue(<@имя_триггера.имя_переменной>,<значение>) - устанавливает значение переменной в кеше.

Show(<@объект>) – делает объект или слой видимым.

Hide(<@объект>) – делает объект или слой скрытым.

Log(<текст_сообщения>) - вывод сообщения локально у пользователя.

sayAll(<@юнит_отправитель>,<текст_сообщения>) - вывод сообщения всем игрокам от имени юнита.

saySide(<@юнит_отправитель>,<текст_сообщения>) - вывод сообщения всей стороне юнита, указанного как отправитель.

sayTeam(<@юнит_отправитель>,<текст_сообщения>) - вывод сообщения в канал группы юнита отправителя.

sayUnit(<@юнит_отправитель>,<@юнит_получатель>,<текст_сообщения>) – вывод сообщения конкретному пользователю или лидеру группы от имени юнита отправителя.

delay(<секунд>) - создает указанную задержку перед выполнением следующей команды.

shell(<команда>) – выполняет одно из перечисленных в константах действий из меню интерфейса.

navTo(<@юнит_получатель>,<@объект_цель>,<действие>) - устанавливает объект в качестве цели навигации для пользователя или группы.

setView(<@юнит_получатель>,<азимут>) – устанавливает азимут обзора юниту.

setDir(<@юнит_получатель>,<азимут>) – устанавливает азимут движения юниту и угол поворота квадрата.

Stop(<@юнит_получатель>) – отменяет азимут обзора и движения юниту.

setLeader(<@юнит>,<@лидер>) – устанавливает лидером группы указанного игрока.

Join(<@юнит>,<@юнит_родитель>) – добавляет юнит в подчинение указанной группы. Если юнит родитель равен isNull, то происходит отсоединение юнита от группы и перенос его в список «без команды».

deleteObject(<@объект>) – удаляет метку или слой с карты.

Block(<@объект>,<статус>) – меняет статус блокировки объекта. Статус может быть «1» (блокировано)или «0» (не блокировано).

varClear(<@имя_триггера>) - очищает среду переменных у триггера.

msgClear() - очищает сообщения на главном экране у пользователя.

update() - вызывает синхронизацию с сервером, чтобы передать сделанные изменения (видимость, статусы, сообщения). У одноразовых триггеров или триггеров с задержкой повтора данная команда вызывается автоматически.

Add(<@объект>,<@множество>) – добавить объект в множество.

Remove(<@объект>,<@множество>) – удалить объект из множества.

Примеры