Описание DG Script

В этом документе описывается возможности DG Script и особенности его использования в МПМ “Былины”. Документ в основном предназначен для начинающих разработчиков зон для МПМ “Былины”, а так же для других МПМ.

История

01.07.2002	0.1	Начальная версия
02.07.2002	0.2	Исправлены орфографические ошибки
19.07.2002	0.3	Добавлено поле char.group Добавлены примеры триггеров Еще раз исправлены орфографические ошибки
11.08.2002	0.4	Код МПМ “Былины” приведен в соответствие с документом Добавлены описания dg_affect и dg_cast Добавлено поле random.num(num) Добавлены примеры триггеров Исправлены множественные ошибки в описаниях функций DG Script set, eval, extract, makeuid, calcuid, unset и др.
24.10.2012	0.45	Обновление и правка переменных
24.08.2015	0.5	Добавлено несколько команд и переменных Правки в некоторых существующих командах/переменных

Содержание

Что такое DG Script

В МПМ “Былины” происходит много различных событий: кто-то вошел в комнату, пользователь ввел команду, кто-то кого-то ударил, кто-то умер и т.д. Для разнообразия игры и внесения дополнительных возможностей многим таким событиям можно назначить обработчики. При этом не нужно каждый раз перекомпилировать саму программу “Былины”, имеется специальный язык сценариев, который позволяет разработчикам новых зон кодировать обработчики и назначать их на различные события в создаваемых ими зонах. При этом не обязательно знать какие-либо языки программирования (хотя их знание может помочь). Язык DG Script специально разработан для решения задачи создания обработчиков событий в МПМ “Былины” и его средства позволяют эффективно и быстро решить задачу.

Триггеры DG SCRIPT

Объекты и триггеры

В МПМ “Былины” можно выделить три основных типа объектов:
1. существа, населяющие мир (как монстры, так и игроки);
2. предметы;
3. комнаты.

Для каждого из типа объектов, существует свое множество событий, для которых можно составить функции-обработчики. Каждая функция-обработчик события называется ***триггером***. Предполагается, что при возникновении события активизируется триггер, выполняет определенные действия и завершается. Однако существуют специальные возможности языка DG Script, которые позволяют затягивать исполнение триггера на большие промежутки времени.

Ниже описаны все типы событий для каждого типа объекта (персонаж, предмет, комната). Каждый триггер (обработчик события) характеризуется конкретным набором параметров, которые необходимо определить при создании триггера. К этим параметрам относятся:
1. Объект, которому назначается триггер, может быть мобом, предметом или комнатой.
2. Тип триггера – определяет событие, на которое активизируется триггер.
3. Числовой аргумент (NArg) – параметр, определяемый создателем триггера. Смысл зависит от типа триггера. Не все триггеры используют этот параметр.
4. Строковый аргумент (Argument) – параметр, определяемый создателем триггера. Смысл зависит от типа триггера. Не все триггеры используют этот параметр.
5. Стандартные локальные переменные – создаются системой автоматически при запуске триггера. Зависят от типа триггера. Подробнее узнать о типах локальных переменных можно в разделе Переменные DG Script.
6. Возвращаемое значение. По умолчанию все триггеры возвращают числовое значение 1. Возвращаемые значения некоторых типов триггеров игнорируются.

Возвращаемым значениям триггеров необходимо уделить немного внимания. Единственное значение, которое может вернуть триггер – значение, устанавливаемое оператором ***return***. Не пытайтесь повлиять на процесс протекания игры, изменяя локальные переменные триггера. Например, заменив для *command* триггера локальные переменные *cmd* и *arg*, вы НЕ СМОЖЕТЕ тем самым изменить введенную команду. Большинство триггеров вызывается в процессе достаточно сложной стандартной процедуры обработки события в мире. Триггер может быть вызван в начале этой процедуры, в конце или середины – это зависит от типа триггера и объекта, к которому он прикреплен. В описании возвращаемого значения для многих триггеров указано **продолжить обработку**. Это обозначает, что в этом случае встроенная стандартная процедура обработки события продолжит свое выполнение и при этом не обязательно действие завершится успешно. В противном случае, обычно выполнение стандартной процедуры прекращается, т.е. действие завершается неуспешно.

Типы триггеров для мобов

Random / Global

Событие:	Периодически, примерно каждые 13 сек. Если для Random триггера не установлен тип Global, для запуска необходимо наличие хотя бы одного PC в зоне с мобом. В противном случае (Global установлен) наличие PC в зоне с мобом не требуется.
NArg:	Вероятность (в процентах), что триггер будет запущен.
Argument:	Не используется
Возвращаемое значение:	Не используется
Локальные переменные:	Нет

Bribe

Событие:	Мобу, к которому “прикреплен” триггер, дали деньги.		
NArg:	Минимальное количество денег, которое нужно дать, чтобы триггер запустился.		
Argument:	Не используется		
Возвращаемое значение:	Не используется		
Локальные переменные:	*actor*	UID	существо, дающее деньги
	amount	число	количество даваемых денег

Memory

Событие:	Мобу, увидел того, кого раньше запоминал. Вызывается как в случае, если моб зашел к жертве в комнату, так и наоборот.		
NArg:	Вероятность (в процентах), что триггер будет запущен.		
Argument:	Не используется		
Возвращаемое значение:	Не используется		
Локальные переменные:	*actor*	UID	идентификатор персонажа из памяти.

Примечание (от автора): Не ясно, почему для случая захода жертвы в комнату к мобу не создается переменной направления входа, как в обычном Greet триггере. То же самое и про вход моба в комнату.

Greet / Greet-All / Greet PC / Greet-All PC

Событие:	К мобу кто-то вошел, а именно Greet – PC или NPC, которого моб видит Greet-All – PC или NPC (моб может не видеть вошедшего) Greet PC – PC, которого моб видит Greet-All PC– PC (моб может не видеть вошедшего)		
NArg:	Вероятность (в процентах), что триггер будет запущен.		
Argument:	Не используется		
Возвращаемое значение:	Используется, но непонятно как :)		
Локальные переменные:	*actor*	UID	идентификатор вошедшего персонажа
	direction	направление	откуда вошел персонаж, если такой переменной нет, то персонаж просто появился в комнате.

Примечание (от автора): на мой взгляд, введение Great PC и Great-All PC лишнее, т.к. сделать проверку на тип вошедшего персонажа в триггере достаточно просто, и зачем нужны дополнительные типы триггеров не очень понятно.

Income / Income PC

Событие:	Моб вошел в комнату / в которой есть хотя бы один видный ему PC.		
NArg:	Вероятность (в процентах), что триггер будет запущен.		
Argument:	Не используется		
Возвращаемое значение:	Не используется		
Локальные переменные:	*actor*	UID	идентификатор последнего PC в комнате (для Income PC)
	direction	направление	откуда вошел моб, если такой переменной нет, то моб просто появился в комнате.

Entry

Событие:	Моб пытается войти в комнату (но еще не вошел).
NArg:	Вероятность (в процентах), что триггер будет запущен.
Argument:	Не используется
Возвращаемое значение:	== 0 – моб не сможет войти в комнату

<> 0 – продолжить обработку
Локальные переменные: Нет

Примечание (от автора): Хоть бы сказали, в какую комнату хочет моб войти. А то вообще не понятно.

Command

Событие:	Кто-то пытается выполнить команду		
NArg:	Способ сравнения команды с текстовым образцом 0 – аргумент - фраза (список фраз) 1 – аргумент - слово (список слов) другой – точное совпадение.		
Argument:	Образец команды. Если образец начинается с символа '*', то триггер будет запускаться на любую команду, независимо от значения NArg.		
Возвращаемое значение:	== 0 – команда не обработана, продолжить стандартную интерпретацию <> 0 – команда обработана		
Локальные переменные:	*actor*	UID	выполняющий команду персонаж
	cmd	строка	команда, на которую сработал триггер
	arg	строка	аргументы команды

Примечание (от автора): Слабо понятно, зачем команду сравнивать с фразой.

Speech

Событие:	Кто-то что-то сказал в комнате с мобом		
NArg:	Способ сравнения сказанного с текстовым образцом 0 – аргумент - фраза (список фраз) 1 – аргумент - слово (список слов) другой – точное совпадение.		
Argument:	Образец сказанного. Если образец начинается с символа '*', то триггер будет запускаться на любую фразу, независимо от значения NArg.		
Возвращаемое значение:	Не используется		
Локальные переменные:	*actor*	UID	выполняющий команду персонаж
	speech	строка	сказанная фраза целиком

Act

TBD (MTRIG_ACT)

Fight

Событие:	Для моба начинает очередной раунд боя (его очередь атаковать)		
NArg:	Вероятность (в процентах), что триггер будет запущен.		
Argument:	Не используется.		
Возвращаемое значение:	1 – моб будет выполнять необходимые действия 0 – моб пропустит раунд		
Локальные переменные:	*actor*	UID	персонаж, с которым дерется моб

Примечание (от автора): Описанная функциональность триггера может отличаться от реализованной в МПМ "Былины".

Damage

Событие:	Моб получил повреждение		
NArg:	Вероятность (в процентах), что триггер будет запущен.		
Argument:	Не используется.		
Возвращаемое значение:	Реально получаемые повреждения. 0 – моб не будет поврежден.		
Локальные переменные:	*damager*	UID	наносящий повреждения персонаж
	amount	число	размер повреждений
	weapon	UID	оружие, которым наносятся повреждения
	skill	строка	заклинание или умение, наносящее повреждения

Примечание (от автора): Описанная функциональность триггера может отличаться от реализованной в МПМ “Былины”.

Hit

Событие:	Моб наносит повреждение противнику		
NArg:	Вероятность (в процентах), что триггер будет запущен.		
Argument:	Не используется.		
Возвращаемое значение:	Не используется.		
Локальные переменные:	*victim*	UID	персонаж, которому наносятся повреждения
	amount	число	размер повреждений
	weapon	UID	оружие, которым наносятся повреждения
	skill	строка	заклинание или умение, наносящее повреждения

Примечание (от автора): Описанная функциональность триггера может отличаться от реализованной в МПМ “Былины”.

Percent

TBD (MTRIG_HITPRCNT)

Receive

Событие:	Мобу пытаются дать предмет.		
NArg:	Вероятность (в процентах), что триггер будет запущен.		
Argument:	Не используется		
Возвращаемое значение:	== 0 – моб не станет брать предмет		
	<> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	идентификатор дающего предмет мобу
	object	UID	передаваемый предмет

Death

Событие:	Моб умер		
NArg:	Не используется		
Argument:	Не используется		
Возвращаемое значение:	== 0 – посмертного крика не будет		
	<> 0 – посмертный крик моба будет слышен		
Локальные переменные:	*actor*	UID	убийца

Примечание (от автора): Формально код рассчитан на вызов триггера при отсутствии убийцы (ну мало ли возможностей самому умереть). Однако вызов процедуры оформлен так, что без убийцы этого не произойдет. Непонятно почему.

Load

Событие: Моб создан и загружен в мир.
NArg: Вероятность (в процентах), что триггер будет запущен
Argument: Не используется
Возвращаемое значение: Не используется
Локальные переменные: Нет

Каст в моба

Событие: На моба кастанули заклинание
NArg: Вероятность (в процентах), что триггер будет запущен
Argument: Не используется
Возвращаемое значение: Не используется
Локальные переменные: actor UID произнесший заклинание
castnum - номер закла из spells.h

Агродействие

Событие: срабатывает при начале боя с мобом (до дамага)
NArg: Вероятность (в процентах), что триггер будет запущен
Argument: Не используется
Возвращаемое значение: Не используется
Локальные переменные: actor UID попытавшийся нанести повреждение

Раунд боя

Событие: срабатывает в бою
NArg: раунд боя на котором сработать
Argument: Не используется
Возвращаемое значение: Не используется
Локальные переменные: actor UID атакуемый моб

Смена времени

Отключен.

Типы триггеров для предметов

Random / Global

Событие: Периодически, примерно каждые 13 сек. Если для Random триггера не установлен тип Global, для запуска необходимо наличие хотя бы одного PC в зоне, в которой находится предмет. В противном случае (Global установлен) наличие PC в зоне не требуется.
NArg: Вероятность (в процентах), что триггер будет запущен
Argument: Не используется
Возвращаемое значение: Не используется
Локальные переменные: Нет

Timer

Событие: Истек таймер предмета
NArg: Не используется
Argument: Не используется
Возвращаемое значение: Не используется

Локальные переменные: Нет

Примечание (от автора): обработка триггера в коде странная, не используйте этот триггер.

Get

Событие:	Предмет поднимают с земли или берут из контейнера		
NArg:	Вероятность (в процентах), что триггер будет запущен		
Argument:	Не используется		
Возвращаемое значение:	== 0 – предмет не возьмется		
	<> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	идентификатор берущего

Command

Событие:	Кто-то пытается выполнить команду рядом с предметом		
NArg:	Битовая маска местонахождения предмета		
	1 – предмет должен находиться в экипировке выполняющего команду		
	2 – предмет должен находиться в инвентаре выполняющего команду		
	4 – предмет должен находиться в одной комнате с выполняющим команду		
Argument:	Образец команды. Если образец начинается с символа ‘*’, то триггер будет запускаться на любую команду, иначе начало введенной команды должно совпадать с этим параметром.		
Возвращаемое значение:	== 0 – команда не обработана, продолжить стандартную интерпретацию		
	<> 0 – команда обработана		
Локальные переменные:	*actor*	UID	выполняющий команду персонаж
	cmd	строка	команда, на которую сработал триггер
	arg	строка	аргументы команды

Wear

Событие:	Кто-то пытается одеть предмет		
NArg:	Не используется		
Argument:	Не используется		
Возвращаемое значение:	== 0 – предмет не будет одет		
	<> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	персонаж, одевающий предмет
	where	число	положение одеваемого предмета

Значения переменной *where* в **Wear** триггере предметов

<для освещения>..... 0
<правый указательный палец> 1
<левый указательный палец>..... 2
<на шее> 3
<на груди> 4
<на теле>..... 5
<на голове>..... 6
<на ногах>..... 7
<на ступнях> 8
<на кистях>..... 9
<на руках>..... 10
<щит> 11
<вокруг тела> 12
<на поясе>..... 13
<на правом запястье>..... 14
<на левом запястье>..... 15
<в правой руке>..... 16
<в левой руке>..... 17
<в руках>..... 18

Remove

Событие:	Кто-то пытается снять предмет		
NArg:	Не используется		
Argument:	Не используется		
Возвращаемое значение:	== 0 – предмет не будет снят		
	<> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	персонаж, снимающий предмет

Drop

Событие:	Кто-то пытается избавиться от предмета (бросить или положить в контейнер)		
NArg:	Вероятность (в процентах), что триггер будет запущен		
Argument:	Не используется		
Возвращаемое значение:	== 0 – предмет не будет брошен		
	<> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	персонаж, избавляющийся от предмета

Give

Событие:	Кто-то пытается отдать предмет		
NArg:	Вероятность (в процентах), что триггер будет запущен.		
Argument:	Не используется		
Возвращаемое значение:	== 0 – предмет не будет отдан		
	<> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	кто пытается отдать предмет
	victim	UID	кому пытаются передать предмет

Load

Событие:	Предмет создан и загружен в мир.
NArg:	Вероятность (в процентах), что триггер будет запущен
Argument:	Не используется
Возвращаемое значение:	Не используется
Локальные переменные:	Нет

Pick

Событие:	Кто-то пытается взломать предмет		
NArg:	Вероятность (в процентах), что триггер будет запущен		
Argument:	Не используется		
Возвращаемое значение:	== 0 – предмет не будет взломан		
	<> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	персонаж, взламывающий предмет

Unlock / Open

Событие:	Кто-то пытается отпереть/открыть предмет
NArg:	Вероятность (в процентах), что триггер будет запущен
Argument:	Не используется

Возвращаемое значение:	== 0 – отпереть/открыть предмет не получится		
	<> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	персонаж, выполняющий действие
	mode	число	режим: 0-“открыть”, 1–“отпереть”.

Примечание (от автора): переменная *mode*, по меньшей мере, странная. Для Open триггеров эта переменная всегда 0, а для Unlock – 1. И зачем нужна такая переменная?

Lock / Close

Событие:	Кто-то пытается запереть/закрыть предмет
NArg:	Вероятность (в процентах), что триггер будет запущен
Argument:	Не используется
Возвращаемое значение:	== 0 – запереть/закрыть предмет не получится
	<> 0 – продолжить обработку

Локальные переменные:	*actor*	UID	персонаж, выполняющий действие
	mode	число	режим: 0-“закрыть”, 1–“запереть”.

Примечание (от автора): переменная *mode,* по меньшей мере, странная. Для Close триггеров эта переменная всегда 0, а для Lock – 1. И зачем нужна такая переменная?

Greet-All PC

Событие:	К предмету, лежащему на полу комнаты, вошел PC (даже невидимый)
NArg:	Вероятность (в процентах), что триггер будет запущен.
Argument:	Не используется
Возвращаемое значение:	Используется, но непонятно как :)

Локальные переменные:	*actor*	UID	идентификатор вошедшего персонажа
	direction	направление	откуда вошел персонаж, если такой переменной нет, то персонаж просто появился в комнате.

Примечание (от автора): триггер сделан по принципу “на безрыбье и рак - рыба”. Неужели нельзя было сделать систему триггеров, аналогичную Greet системе для мобов. Примечание к тому набору триггеров также относится и к этому триггеру.

Типы триггеров для комнат

Reset

Событие:	Зона, в состав которой входит комната, перегружается
NArg:	Вероятность (в процентах), что триггер будет запущен
Argument:	Не используется
Возвращаемое значение:	Не используется
Локальные переменные:	Нет

Random / Global

Событие:	Периодически, примерно каждые 13 сек. Если для Random триггера не установлен тип Global, для запуска необходимо наличие хотя бы одного PC в зоне, к которой принадлежит комната. В противном случае (Global установлен) наличие PC в зоне с комнатой не требуется.
NArg:	Вероятность (в процентах), что триггер будет запущен.
Argument:	Не используется

Возвращаемое значение: Не используется
Локальные переменные: Нет

Enter / Enter-PC

Событие:	В комнату пытаются войти [кто угодно / PC]		
NArg:	Вероятность (в процентах), что триггер будет запущен.		
Argument:	Не используется		
Возвращаемое значение:	== 0 – войти нельзя <> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	идентификатор вошедшего персонажа
	direction	направление	откуда вошел персонаж, если такой переменной нет, то персонаж просто появился в комнате.

Command

Событие:	Кто-то пытается выполнить команду в комнате		
NArg:	Способ сравнения команды с текстовым образцом 0 – аргумент - фраза (список фраз) 1 – аргумент - слово (список слов) другой – точное совпадение.		
Argument:	Образец команды. Если образец начинается с символа ‘*’, то триггер будет запускаться на любую команду, независимо от значения NArg.		
Возвращаемое значение:	== 0 – команда не обработана, продолжить стандартную интерпретацию <> 0 – команда обработана		
Локальные переменные:	*actor*	UID	выполняющий команду персонаж
	cmd	строка	команда, на которую сработал триггер
	arg	строка	аргументы команды

Speech

Событие:	Кто-то что-то сказал в комнате		
NArg:	Способ сравнения команды с текстовым образцом 0 – аргумент - фраза (список фраз) 1 – аргумент - слово (список слов) другой – точное совпадение.		
Argument:	Образец команды. Если образец начинается с символа ‘*’, то триггер будет запускаться на любую команду, независимо от значения NArg.		
Возвращаемое значение:	Не используется		
Локальные переменные:	*actor*	UID	выполняющий команду персонаж
	speech	строка	сказанная фраза целиком

Drop

Событие:	Кто-то в комнате пытается бросить предмет на землю		
NArg:	Вероятность (в процентах), что триггер будет запущен		
Argument:	Не используется		
Возвращаемое значение:	== 0 – предмет не будет брошен <> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	персонаж, бросающий предмет
	object	UID	бросаемый объект

Примечание (от автора): Триггер проверяется в двух случаях: бросают деньги и бросают предмет. Деньги пропадают бесследно в случае возвращения 0. Триггер никак не связан с переносом объекта в комнату другим способом. А упасть на пол можно столькими способами. Есть предложение разместить этот триггер в функции obj_decay() или obj_to_room(). Короче, странный триггер, а можно было бы такого сделать.

Pick

Событие:	Кто-то пытается взломать дверь в комнате		
NArg:	Вероятность (в процентах), что триггер будет запущен		
Argument:	Не используется		
Возвращаемое значение:	== 0 – дверь не будет взломана		
	<> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	персонаж, взламывающий дверь
	direction	направление	направление взламываемой двери

Unlock / Open

Событие:	Кто-то пытается отпереть/открыть дверь в комнате		
NArg:	Вероятность (в процентах), что триггер будет запущен		
Argument:	Не используется		
Возвращаемое значение:	== 0 – отпереть/открыть дверь не получится		
	<> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	персонаж, выполняющий действие
	direction	направление	направление двери
	mode	число	режим: 0-“открыть”, 1–“отпереть”.

Примечание (от автора): переменная *mode,* по меньшей мере, странная. Для Open триггеров эта переменная всегда 0, а для Unlock – 1. И зачем нужна такая переменная?

Lock / Close

Событие:	Кто-то пытается запереть/закрыть дверь		
NArg:	Вероятность (в процентах), что триггер будет запущен		
Argument:	Не используется		
Возвращаемое значение:	== 0 – запереть/закрыть дверь не получится		
	<> 0 – продолжить обработку		
Локальные переменные:	*actor*	UID	персонаж, выполняющий действие
	direction	направление	направление двери
	mode	число	режим: 0-“закрыть”, 1–“запереть”.

Примечание (от автора): переменная *mode,* по меньшей мере, странная. Для Close триггеров эта переменная всегда 0, а для Lock – 1. И зачем нужна такая переменная?

Общие правила для триггеров

В этом разделе будут даны советы как нужно и как не нужно использовать триггеры. Будут приведены самые распространенные ошибки при использовании триггеров и даны некоторые разъяснения по поводу взаимодействия триггеров разного типа друг с другом.

1. Триггеры одного типа для объекта.

В принципе, объекту можно назначить несколько разных триггеров для обработки одного и того же события (одинаковые триггеры назначить нельзя). Этого делать не рекомендуется, однако иногда такое решение позволяет требуемую функциональность. Поиск триггера для обработки события происходит в порядке, обратном их назначению объекту. Если подходящий триггер найден, начинается проверка его параметров (шансы вызова, аргументы и т.д.). В случае успешной проверки – триггер вызывается, иначе пропускается, и поиск триггера продолжается дальше. Все триггеры после своего завершения (с любым возвращенным значением) прекращают дальнейший поиск, за исключением MOB_COMMAND и OBJ_COMMAND (обратите внимание, что WLD_COMMAND не является исключением). Для этих триггеров происходит дальнейший поиск обработчика, если вызванный обработчик вернул 0, т.е. команда не обработана. В результате могут быть проверены все возможные обработчики команд мобов и объектов.

Примечание (от автора): непонятно почему такой возможности лишены WLD_COMMAND триггеры.

К обсуждаемому вопросу косвенно относится проблема идентичных триггеров для разных объектов, расположенных в одном месте. Те же самые Command триггеры проверяются только до первого совпадения и если в комнате расположены несколько одинаковых мобов с триггерами Command, то выполнится только 1 триггер. Для триггеров типа MOB_GREET и OBJ_GREET проверяются триггеры всех объектов, и результат получается логическим произведением возвращаемых значений.

2. Запуск триггера

Триггер может быть на запущен не только по причине несовпадения его параметров. Еще необходимо, чтобы этот триггер не выполнялся для данного объекта. Для MOB-триггеров еще важно состояние моба, которому назначен триггер. Т.о. можно сформулировать такие правила:

1. Триггер не будет запущен, если для данного объекта он уже выполняется;
2. Для MOB-триггеров GREET, MEMORY и SPEECH моб должен бодрствовать.

3. Взаимодействие триггеров

Многие триггеры при обработке различных ситуаций работают вместе.
Например, при передаче объекта сначала вызывается OBJ_GIVE триггер и, в случае успешного его выполнения, вызывается MOB_RECEIVE триггер. Предмет будет передан, если MOB_RECEIVE триггер завершится успешно.

4. Приоритеты COMMAND триггеров

WORLD_COMMAND ←высший
MOB_COMMAND
OBJ_COMMAND
<стандартные команды>
<социалы> ← низший

5. Задержки в Death триггере

Никогда не используйте задержки в DEATH триггере. По команде паузы (*wait*) выполнение триггера прервется, чтобы продолжиться позднее. Моб будет уничтожен (все же это его смерть) вместе со сценарием и триггером соответственно. Т.о. вы никогда ничего не выполните после паузы в Death триггерах. Такое поведение может быть вызвано не только явными командами wait, но и их скрытыми вариантами.

Переменные DG Script

Как все языки программирования DG Script обеспечивает возможность работать с переменными различных типов. Кроме того, в DG Script определяется три класса переменных, которыми определяется область видимости и время жизни переменной.

Классы переменных и контекст сценария

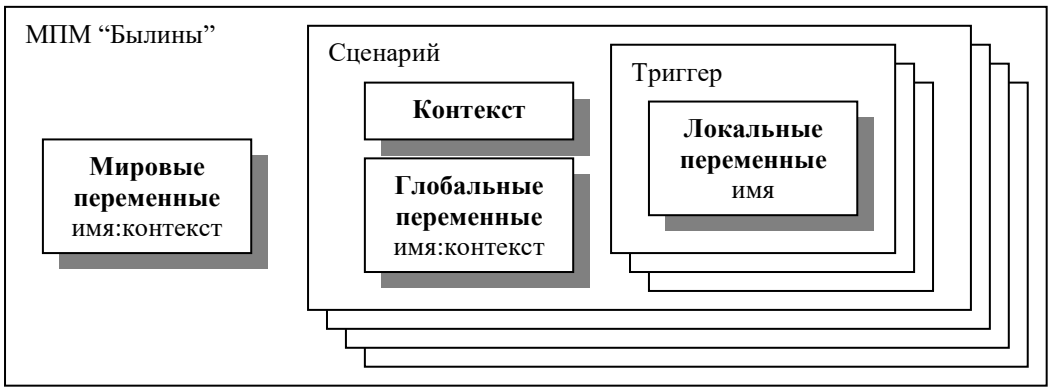
DG Script определяет три класса переменных: ***локальные, глобальные, мировые.***
Объект может содержать несколько триггеров обработки различных событий. Множество триггеров для объекта называется ***сценарием***. Сценарий это не просто объединение триггеров, он еще обладает некоторым ***контекстом*** и определяет время жизни глобальных переменных. Если у объекта нет ни одного триггера, у него нет сценарий со всеми вытекающими последствиями.

Локальные переменные принадлежат конкретному триггеру в сценарии объекта и существуют только во время выполнения триггера. Значения переменных между вызовами триггера не сохраняются. На эти переменные (в отличие от глобальных и мировых) не влияет контекст сценария. Все параметры, которые устанавливает программа при вызове триггера, передаются как локальные переменные. Подробнее об этих локальных переменных можно посмотреть в разделе Триггеры DG Script. Количество и размер данных локальных переменных ограничен только объемом памяти компьютера, на котором работает МПМ “Былины”.

Сценарий объекта может содержать ***глобальные переменные*** и в каждый конкретный момент времени обладать определенным ***контекстом***. Глобальные переменные сохраняют свое значение между вызовами отдельных триггеров сценария, доступны всем триггерам данного сценария и также доступны из триггеров других (!) объектов. Идентификация глобальных переменных сценария осуществляется не только по имени, но и по значению контекста. Глобальные переменные автоматически уничтожаются при уничтожении сценария. Контекст сценария сохраняется в течение жизни сценария объекта, он одинаков для всех триггеров данного сценария и может быть изменен специальными командами DG Script.

Мировые переменные очень похожи на глобальные переменные сценария, за исключением того, что они не принадлежат ни одному сценарию и время их жизни совпадает со временем жизни мира. При этом, как и для глобальных переменных, есть возможность создания и уничтожения переменных из триггеров объектов.

Взаимодействие различных классов переменных, сценариев и контекста представлено на рисунке.



Локальные переменные сценария идентифицируются только по имени. Происходит сравнение имен с учетом регистра. Контекст сценария позволяет индексировать мировые переменные и глобальные переменные сценария (на локальные переменные контекст сценария не влияет). Каждую переменную перечисленных классов идентифицирует пара *имя:контекст*. Общим контекстом называется значение контекста равное 0. При поиске переменной проверяется совпадение и имени (сравнение с учетом регистра), и контекста переменной. Имя должно совпасть обязательно, а контекст проверяется сложнее. Сначала ищется переменная с запрошенным контекстом:
а) Если такой переменной не оказывается и запрошенный контекст == 0, то сообщается, что переменной нет.
б) Если такой переменной не оказывается и требуемый контекст <> 0, то производится попытка поиска переменной в общем контексте (контекст = 0).

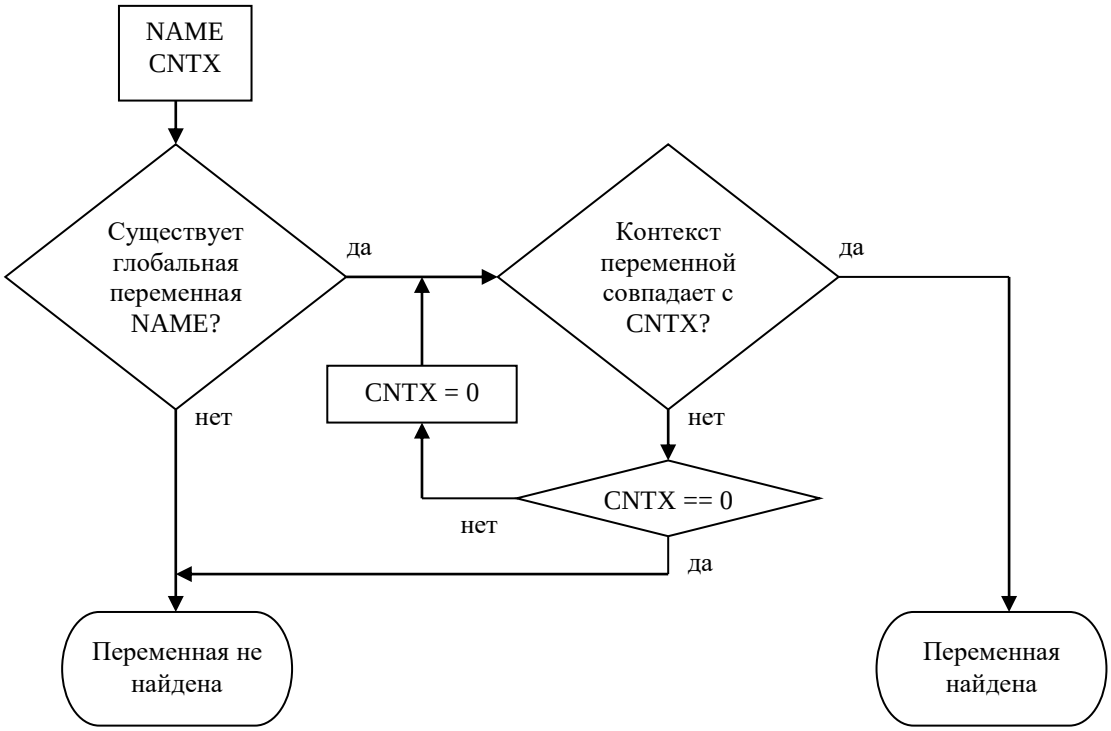


Рисунок. Алгоритм поиска глобальной/мировой переменной.

Область видимости переменных, если явным образом не указан тип доступа, следующая:
1. Поиск локальной переменной триггера
2. Поиск глобальной переменной сценария с текущим контекстом сценария
3. Поиск мировой переменной с текущим контекстом сценария

Типы данных DG Script

Значения всех переменных в DG Script – текстовые строки. Но в зависимости от оператора, содержимого строки и т.д. можно выделить ***типы данных*** переменных DG Script.

Строка

Строка – всеобъемлющий тип данных DG Script. Как было сказано выше, значения всех переменных представляют собой текстовые строки. При этом в этих строках могут содержаться разделители, знаки арифметических действий, различные непечатные символы и т.д. Т.е. переменная DG Script может принимать практически любое значение и значение любой переменной можно рассматривать как текстовую строку и выполнять с ней операции, как с текстовой строкой.

Число

Текстовая строка, представляет собой запись целого со знаком, является *числом*. Числа достаточно часто используются в DG Script, поэтому имеет смысл выделить их в отдельный тип данных.

Направление

Некоторым триггерам передается параметр-направление. Тип данных *направление* – это строка, которая представляет собой одно из слов: *north, east, south, west, up, down*.

Список

Текстовую строку можно рассматривать как *список*. Элементами этого списка являются части строки, разделенные пробелами. Т.о. просто слово является списком из одного элемента. Пустая строка – пустой список.

Уникальный идентификатор (UID)

Все объекты мира (персонажи, мобы, предметы, комнаты) имеют свой уникальный идентификатор, по которому можно однозначно установить объект, о котором идет речь. DG Script позволяет переменным иметь значениями уникальные идентификаторы. Такой тип данных называется *UID*. ВНИМАНИЕ: UID это не ЧИСЛО. UID это особое текстовое представление уникального идентификатора (ссылки) на объект мира. Однако существуют специальные возможности преобразования UID в число и обратно.

Значения переменных

Переменные заменяются на их значения в процессе “подстановки значений” (см. раздел Команды DG Script). На место переменной в строку вставляется ее значение. Для того чтобы текст был интерпретирован как имя переменной, его необходимо заключить в символы ‘%’.

Существует три формы представления переменной.
1. Простая переменная: ***%name%***. Форма используется для получения значения переменной *name*.
2. Сложная переменная: ***%name.field%***. Существует большое количество переменных, для которых определены т.н. *поля*. Форма используется для доступа к полю *filed* переменной *name*. При этом для получения значения поля переменной, возможно, необходимо произвести различные преобразования.
3. Параметризованная переменная: ***%name.field(param)%***. Если при вычислении поля необходимо знать дополнительные параметры, то используется эта форма представления переменной. Тип значения *param* зависит от конкретного поля *filed* переменной *name*. Особенность такой формы записи переменной в том, что поле *param* может быть в свою очередь выражено переменной.

Примеры:

%actor%	получение UID персонажа actor
%actor.name%	получение имени персонажа actor
%actor.hitp(100)%	установка здоровья 100 hp
%actor.hitp(+%random.100%)%	увеличение здоровья на случ. величину [1..100]

Любая переменная может быть использована в своей простой форме, т.е. *%name%*.
Если переменная используется в сложной или параметризованной форме, то значение переменной *name* должно быть одним из объектов мира или *name* должна быть встроенной переменной. Поиск объекта мира по значению переменной осуществляется в следующем порядке:
1. Для MOB-триггеров: предмет в экипировке, предмет в инвентаре, персонаж в комнате, предмет в комнате, персонаж, предмет, комната.
2. Для OBJ-триггеров: персонаж, предмет, комната.
3. Для WLD-триггеров: персонаж, предмет, комната.
Очевидно, что если значением переменной будет UID, то поиск будет более предсказуемым и быстрым. Предпочтительнее использовать UID для ссылки на объект мира, чем имя объекта.

Если переменная не была найдена, то она заменяется на пустую строку (стирается из текста). Ниже по тексту пустая строка будет называться ***nil***.
В нижеследующих разделах перечисляются все доступные в DG Script переменные.

Автозамена команд

В сценариях DG Script существуют дополнительные команды, недоступные для игроков. При этом существуют модификации таких команд для разного типа объектов (мобы/предметы/комнаты). Для облегчения написания триггеров в DG Script поддерживается механизм автозамены команд, который

каждую команду-псевдопеременную заменит на команду, подходящую данному типу триггера. Список этих команд приведен в таблице.

Псевдопеременная	Команда для мобов	Команда для предметов	Команда для комнат
%send%	msend	osend	wsend
%echo%	mecho	oecho	wecho
%echoaround%	mechoaround	oechoaround	wechoaround
%door%	mdoor	odoor	wdoor
%force%	mforce	oforce	wforce
%load%	mload	oload	wload
%purge%	mpurge	opurge	wpurge
%teleport%	mteleport	oteleport	wteleport
%damage%	mdamage	odamage	wdamage
%featturn%	mfeatturn	ofeatturn	wfeatturn
%skillturn%	mskillturn	oskillturn	wskillturn
%skilladd%	mskilladd	oskilladd	wskilladd
%spellturn%	mspellturn	ospellturn	wspellturn
%spellturntemp%	mspellturntemp	ospellturntemp	wspellturntemp
%spelladd%	mspelladd	ospelladd	wspelladd
%spellitem%	mspellitem	ospellitem	wspellitem
%portal%	mportal	oportal	wportal

%send% Команда отправляет игроку сообщение. Синтаксис %send% %кому.id% <сообщение>
%echo% Текст в клетку. Синтаксис: %echo% <сообщение>
%echoaround% Текст в клетку. Синтаксис: %echoaround% %комуневидносообщение.id% <сообщение>
%force% Заставить моба или игрока выполнить команду: Синтаксис %force% %кого.id% командамада
%door% Открывает проход. Синтаксис %door% <номер комнаты откуда> < географическое направление куда> <параметр> ("purge", "description", "flags", "key", "name", "room", "lock")

Purge – удаляет выход
Description – Новое описание выхода
Flags – флаги
a - У выхода есть дверь открывается/закрывается.
b - Дверь закрыта.
c - Дверь заперта.
d - Замок нельзя взломать.
e - Дверь секретная
Key – номер ключа чтоб отпереть
Name – имя двери
Room – в какой номер клетки делать выход
Lock – сложность замка

%load% Загрузить предмет или моба в клетку. Синтаксис: %load% <mob|obj> <номер клетки>
%purge% Уничтожить предмет или моба. Синтаксис %purge% <id предмета>
%teleport% Переместить игрока или моба. Синтаксис %teleport% %кого.id% <номер клетки> не обязательные параметры: [все (вместе с чармисами) horse (вместе с лошадью)] Внимание! Для mteleport еще доступен необязательный параметр followers, телепортирует всех следующих за тем на кого сработала команда.
%damage% Нанести повреждение игроку или мобу. Синтаксис %damage% %кому.id% <количество повреждений>
%featturn% Установить способность. Синтаксис %featturn% %кому.id% <название.способности> <set или clear>. Соответственно set – установить, clear – убрать способность.
%skillturn% Установить умение. Синтаксис %skillturn% %кому.id% <название.умения> <set или clear>. Соответственно set – установить, clear – убрать умение.
%skilladd% Изменить текущий параметр умения игрока. Синтаксис %skilladd% %кому.id% <название.умения> <+- % умения>
%spellturn% Научить игрока заклинанию. Синтаксис %spellturn% %кому.id% <название.заклинания> <set или clear>. Соответственно set – установить, clear – убрать заклинание.
%spellturntemp% Временно научить игрока заклинанию. Синтаксис %spellturntemp% %кому.id% <название.заклинания> <время в секундах>. По окончании указанного времени заклинание забывается и удаляется из мема.
%spelladd% - не используется.
%spellitem% - не используется

%portal% Команда открывает пентаграмму из текущей комнаты в заданную комнату. Синтаксис wportal <номер комнаты> <длительность портала> (пока реализована только команда wportal)

wzoneecho Отправляет сообщение всем игрокам в зоне. Синтаксис wzoneecho <номерзоны> <сообщение>

Встроенные переменные

Переменная	Тип	Описание	Результат
self	UID	Получение владельца триггера	Владелец триггера
exist.mob(vnum)	число	Проверка существования в мире моба vnum	1 – моб существует 0 – странный результат nil – моба не существует
exist.obj(vnum)	число	Проверка существования в мире (в игре в текущий момент и на ренте) предмета vnum	1 – предмет существует 0 – странный результат nil – предмета не существует
world.curmobs(vnum)	число	Количество vnum-мобов в мире в текущий момент	Количество мобов nil – моба нет
world.curobj(vnum) **world.curobjs**(vnum)	число	Количество vmun-предметов в мире (в игре в текущий момент и на ренте)	Количество предметов nil – предметов нет если качество нерушимо - 0
world.gamemobs(vnum)	число	Количество vmun-мобов + их умертвий в мире (в текущий момент)	Количество мобов nil – мобов нет
world.gameobjs(vnum)	число	Количество vmun-предметов в мире (в игре в текущий момент)	Количество предметов nil – предметов нет
world.maxobj(vnum) или **world.maxobjs**(vnum)	число	Максимальное количество vmun-предметов в мире (проставляется в свойствах предмета)	Количество предметов nil – предметов нет если качество нерушимо - 9999999
world.people(vmun)	число	Количество персонажей в комнате vnum (PC, NPC и др)	Количество персонажей -1 если комнаты не существует
world.zreset(vnum)	-	Вызов процедуры ресета зоны vnum	nil
world.mob(vnum)	число	Получение численного значения UID персонажа VNUM	Численное значение UID
world.obj(vnum)	число	Получение численного значения UID пердмета VNUM	Численное значение UID
world.room(vnum)	число	Получение численного значения UID комнаты VNUM	Численное значение UID
weather.moon	число	Возраст луны	Возраст в днях
weather.season	строка	Время года	зима/весна/лето/осень
weather.sky **weather.sky**(vnum)	число	Облачность в мире Облачность в комнате VNUM	0 – облачно 1 – пасмурно 2 – тяж. тучи 3 – ясно
weather.sunlight	строка	Время суток	ночь/закат/день/рассвет
weather.temp	число	Температура на дворе	Температура
weather.type **weather.type**(vnum)	строка	Множество букв, описывающее текущую погоду в мире или в комнате VNUM соответственно	a – резкое похолодание b – резкое потепление c – моросящий дождь d – дождь e – льет как из ведра f – дождь с градом g – снежок h – снегопад i – валит снег j – ветерок k – умеренный ветер l – сильный ветер
time.hour	число	Игровое время, час	Игровой час
time.day	число	Игровое время, день	Игровой день
time.month	число	Игровое время, месяц	Игровой месяц
time.year	число	Игровое время, год	Игровой год

Переменная	Тип	Описание	Результат
date.minute	число	Реал. время, минута	Реал. минут
date.hour	число	Реал. время, час	Реал. часов
date.day	число	Реал время, день месяца	Реальный день месяца
date.month	число	Реал. время, месяц	Реальный месяц
date.year	число	Реал. время, год	Реальный год
date.unix	число	Юниксовый формат реал. времени	Количество секунд с 1970 года
date.wday	число	Номер реал. дня недели	0 – воскресенье … 6 – суббота
date.yday	число	Порядковый номер реал. дня в году	Число от 1 до 366
random.char **random.pc** **random.npc**	UID	Случайный выбор персонажа в комнате. В поиск не включаются self, NOHASSLE и невидимые для моба/объекта персонажи. char – выбор из всех pc – выбор только из pc npc – выбор только из npc	Случайно выбранный персонаж
random.num **random.num**(num)	число	Возвращает случайное число [1,num] Вторую форму удобно использовать, когда значение num заранее не известно.	Случайное число

Примечание: встроенные переменные *exist, world, time, date, weather* и *random* в простой форме НЕ_СУЩЕСТВУЮТ.

Текстовая обработка

Значение любой переменной является текстовой строкой, и к ней может быть применены операции текстовой обработки. Значение самой переменной не изменяется.

Операция	Тип	Действие	Результат
var.**strlen**	число	Вычисление длины строки	Длина строки
var.**trim**	строка	Удаление начальных и конечных пробелов	Урезаная строка
var.**contains**(str)	число	Проверка подстроки	1 – str является подстрокой значения переменной var 0 – str не является подстрокой значения переменной var
var.**car**	строка	Выделение первого слова строки	Первое слово
var.**cdr**	строка	Выделение части строки после первого слова.	Остаток строки
var.**words**	число	Определение количества слов в строке (элементов с списке)	Количество слов
var.**words**(n)	строка	Получение n-ого слова строки	Слово
var.**mudcommand**	строка	Получение полной версии команды MUD по значению переменной var. Ищется стандартная MUD команда, аббревиатурой которой является var.	Команда MUD или nil

Поля переменных моба

Операция	Тип	Действие	Результат
char.**iname**	строка	Имя (именительный падеж)	Имя
char.**rname**	строка	Имя (родительный падеж)	Имя
char.**dname**	строка	Имя (дательный падеж)	Имя
char.**vname**	строка	Имя (винительный падеж)	Имя

Операция	Тип	Действие	Результат
char.**tname**	строка	Имя (творительный падеж)	Имя
char.**pname**	строка	Имя (предложный падеж)	Имя
char.**name**	строка	Короткое описание (если есть), иначе имя	Имя
char.**id**	число	Получение численного значения ID персонажа char.	Численное значение ID
char.**uniq**	число	Получение численного значения UID персонажа char.	Численное значение UID
char.**alias**	строка	Имя	Имя
char.**level**	число	Уровень персонажа	Уровень
char.**remort**	число	Количество перевоплощений	Морты
char.**hitp**	число	Получение количества hp	Текущее значение hp
char.**hitp**(num)	число	Изменение hp. Формат num: num – установить значение в num +num – увеличить значение на num -num – уменьшить значение на num	Новое значение hp
char.**maxhitp**	число	Получение максимального количества hp	Максимальное количество hp
char.**mana**	число	Получение количества маны	Текущее количество маны
char.**mana**(num)	число	Изменения маны. Формат num: num – установить значение в num +num – увеличить значение на num -num – уменьшить значение на num	Новое значение маны
char.**maxmana**	число	Получение максимального количества маны	Максимальное количество маны
char.**move**	число	Получение количества энергии	Текущее количество энергии
char.**move**(num)	число	Изменение энергии. Формат num: num – установить значение в num +num – увеличить значение на num -num – уменьшить значение на num	Новое значение энергии
char.**maxmove**	число	Получение максимального количества энергии	Максимальное количество энергии
char.**age**	число	Возраст персонажа	Возраст (лет)
char.**align**	число	GET_ALIGNMENT	
char.**religion**	число	Религия персонажа	0 – язычник 1 – христианин
char.**gold**	число	Получение количества денег	Количество денег
char.**gold**(num)	число	Установка количества денег Формат num: число – установить значение в num +число – увеличить значение на num -число – уменьшить значение на num	Новое количество денег
char.**bank**	число	Получение количества денег в банке	Количество денег в банке
char.**bank**(num)	число	Установка количества денег в банке Формат num: num – установить значение в num +num – увеличить значение на num -num – уменьшить значение на num	Новое количество денег в банке
char.**exp**	число	Получение опыта	Текущий опыт
char.**exp**(num)	число	Изменение опыта. Формат num: num – установить значение в num +num – увеличить значение на num -num – уменьшить значение на num	Новый опыт
char.**sex**	число	Пол персонажа	0 – средний 1 – мужской 2 – женский 3 – мн. число
char.**clan**	строка	Название клана (в нижнем регистре)	Название клана персонажа
char.**clanrank**	число	Положение в клане	0 – Гость 1 – Отрок 2 – Муж

Операция	Тип	Действие	Результат
			…
char.**g**	строка	о//а/и	Суффикс
char.**u**	строка	ось/ся/ась/ись	Суффикс
char.**w**	строка	ое/ый/ая/ые	Суффикс
char.**q**	строка	ло//ла/ли	Суффикс
char.**y**	строка	ло/ел/ла/ли	Суффикс
char.**a**	строка	о//а/ы	Суффикс
char.**weight**	число	Вес персонажа	Вес персонажа
char.**canbeseen**	число	Проверка видит ли персонаж self, персонаж char.	0 – self не видит персонажа 1 – self видит персонажа или self не моб
char.**class**	число	Класс персонажа	0 – CLERIC 1 – BATTLEMAGE 2 – THIEF 3 – WARRIOR 4 – ASSASINE 5 – GUARD 6 – CHARMMAGE 7 – DEFENDERMAGE 8 – NECROMANCER 9 – PALADINE 10 – RANGER 11 – SMITH 12 – MERCHANT 13 – DRUID
char.**race**	число	Раса персонажа	0 – SEVERANE 1 – POLANE 2 – KRIVICHI 3 – VATICHI 4 – VELANE 5 – DREVLANE
char.**fighting**	UID	Получение противника в бою	Противник или nil
char.**is_killer**	число	Флаг душегуба	0 – игрок душегуб 1 – игрок не душегуб
char.**is_thief**	число	Флаг вора (PLR_THIEF)	0 – игрок не имеет флага 1 – игрок имеет флаг
char.**agressor**	число	Проверка на агробд	0 – нет агробд >0 – vnum комнаты нападения
char.**rentable**	число	Проверка на боевые действия	0 – не может уйти на постой 1 – может уйти на постой
char.**riding**	UID	Определение лошади	Лошадь или nil
char.**ridden_by**	UID	Определение наездника	Наездник или nil
char.**vnum**	число	vnum моба. –1 для PC	VNUM
char.**str**	число	Врожденная сила персонажа	Сила
char.**str**(num)	число	Изменение врожденной силы num – установить силу равную num +num – увеличить силу на num -num – уменьшить силу на num	Сила
char.**stradd**	число	Добавочная сила персонажа	Сила
char.**int**	число	Врожденный ум персонажа	Ум
char.**int**(num)	число	Изменение врожденного ума	Ум
char.**intadd**	число	Добавочный ум персонажа	Ум
char.**wis**	число	Врожденная мудрость персонажа	Мудрость
char.**wis**(num)	число	Изменение врожденной мудрости	Мудрость
char.**wisadd**	число	Добавочная мудрость персонажа	Мудрость
char.**dex**	число	Врожденная ловкость персонажа	Ловкость
char.**dex**(num)	число	Изменение врожденной ловкости	Ловкость
char.**dexadd**	число	Добавочная ловкость персонажа	Ловкость
char.**con**	число	Врожденное тело персонажа	Телосложение
char.**con**(num)	число	Изменение врожденного тела	Телосложение

Операция	Тип	Действие	Результат
char.**conadd**	число	Добавочное тело персонажа	Телосложение
char.**cha**	число	Врожденное обаяние персонажа	Обаяние
char.**cha**(num)	число	Изменение врожденного обаяния	Обаяние
char.**chaadd**	число	Добавочное обаяние персонажа	Обаяние
char.**size**	число	Врожденный размер персонажа	Размер
char.**size**(num)	число	Изменение врожденного размера	Размер
char.**sizeadd**	число	Добавочный размер персонажа	Размер
char.**room**	число	Получение комнаты, в которой находится персонаж	RNUM комнаты
char.**realroom**	число	VNUM комнаты, в которой находится персонаж	VNUM комнаты
char.**loadroom**	число	Получение загрузочной комнаты	VNUM комнаты
char.**loadroom**(vnum)	число	Установка загрузочной комнаты	VNUM комнаты
char.**skill**(str)	число	Уровень умения str у персонажа. Str – название умения	Уровень владения умением
char.**spellcount**(str)	число	Количество у персонажа выученных заклинаний str. Str – имя заклинания	Количество заклинаний
char.**spelltype**(str)	число	Тип запоминания заклинания str у персонажа. Флаги из массива SplKnw. Str – имя заклинания	Пустая строка – не выучено 1 - SPELL_KNOW 2 - SPELL_TEMP 4 - SPELL_POTION 8 - SPELL_WAND 16 - SPELL_SCROLL 32 - SPELL_ITEMS 64 - SPELL_RUNES Если надо проставить несколько флагов одновременно - то надо вычислить сумму флагов и проверить ее.
char.can_get_spel(str)	строка	возвращает значение если персонаж может выучить соотв. закл по мортам и левелу. Соответствующие поля есть для skill и feat	1 –может 0 – не может
char.**quested**(num)	число	Проверка на то, выполнял ли персонаж квест номер num.	1 – выполнял 0 – не выполнял
char.**setquest**(num[txt])	число	Установка признака выполнения квеста номер num	Если первым знаком перед txt поставить @ то признак выполнения не будет удален при реморте.
char.**getquest**(num)	строка	Возвращает пометки, оставленные при setquest в поле txt квеста num	Пометка о выполнении квеста
char.**unsetquest**(num)	число	Отмена признака выполнения квеста номер num	1 – выполнял
char.**eq**(pos)	UID	Получение предмета экипировки. pos – позиция (текст или номер позиции)	Предмет или nil
char.**haveobj**(obj)[1] char.**haveobjs**(obj)	число	Несет ли персонаж предмет obj? obj может быть vnum или именем предмета.	1 – несет 0 – не несет
char.**objs**	список из UID	Вещи в инвентаре персонажа.	Список вещей
char.**varexists**(name)[2]	число	Проверяет у сценария объекта char наличие глобальной переменной name.	1 – есть 0 – нет
char.**position**	число	Возвращает положение персонажа	0 – dead

Операция	Тип	Действие	Результат
			1 – mortally wounded 2 – incapacitated 3 – stunned 4 – sleeping 5 – resting 6 – sitting 7 – fighting 8 – standing
char.**position**(pos)	-	Устанавливает позицию персонажа. Не действует на богов и т.д.	-
char.**wait**	число	Возвращает лаг персонажа в **пульсах**. (один раунд=50 пульсов??)	Лаг
char.**wait**(pause)	-	Устанавливает лаг в **раундах**. Не действует на богов и т.д.	-
char.**affect**(name)	число	Проверяет наличие на персонаже аффекта name.	1 – аффект есть 0 – аффекта нет
char.**affected_by**(name)	число	Проверяет наличие на персонаже аффекта от заклинания *name*.	1 – аффект есть 0 – аффекта нет
char.**leader**	UID	Возвращает лидера для char.	Лидер или nil
char.**people**	UID	Первый персонаж в комнате с char.	Персонаж или nil
char.**next_in_room**	UID	Следующий за char персонаж в комнате	Персонаж или nil
char.**all** char.**char** char.**pc** char.**npc** char.**group** char.**attackers**	список из UID	Список персонажей в комнате: all – все в комнате char – все PC и чармисы pc – все PC npc – все NPC (мобы не чармисы) group – список группы, в которой состоит char. Начинается с лидера, включает всех последователей. attackers – список персонажей, атакующих char.	Список персонажей, может быть пустым
char.**isalliance**(name)	строка	Проверяет нахождение дружины (name) в состоянии альянса с дружиной игрока.	1 – да 0 - нет
char.**global**(name) [2]	строка	Поиск и замена глобальной переменной другого сценария. При поиске переменной используется текущий контекст сценария char.	Значение глобальной переменной сценария char
char.**var** [2]	строка	Поиск и замена глобальной переменной другого сценария. При поиске переменной используется текущий контекст сценария self.	Значение глобальной переменной сценария char
char.**restore**	-	Рестор hp и мувов, кличей и др.	-
char.**dispel**	-	Снятие всех аффектов с персонажа	-
char.**owner**	UID	Используется для привязки предмета к чару - другие его даже взять не смогут.	
char.**can_get_feat**(способность)	строка	Сообщает доступна ли данному персонажу способность	1 – да 0 - нет
char.**feat**(способность)	строка	Сообщает изучена ли способность у игрока	1 – да 0 - нет

[1] Данная операция имеет ошибку в коде. В принципе невозможно задать имя объекта в виде “2.меч” т.к. 2 будет воспринята как признак vnum.
[2] Непонятно, почему такого поля нет у других типов объектов

Поля переменных предмета

Операция	Тип	Действие	Результат

Операция	Тип	Действие	Результат
obj.**iname**	строка	Имя (именительный падеж)	Имя
obj.**rname**	строка	Имя (родительный падеж)	Имя
obj.**dname**	строка	Имя (дательный падеж)	Имя
obj.**vname**	строка	Имя (винительный падеж)	Имя
obj.**tname**	строка	Имя (творительный падеж)	Имя
obj.**pname**	строка	Имя (предложный падеж)	Имя
obj.**name**	строка	Имя	Имя
obj.**id**	число	Получение численного значения UID предмета obj.	Численное значение UID
obj.**shortdesc**	строка	Короткое описание	Описание
obj.**vnum**	число	vnum предмета	VNUM
obj.**type**	число	Тип предмета	Код типа предмета
obj.**timer**	число	Таймер предмета	Таймер
obj.**val0**	число	Параметр предмета 0	Значение параметра
obj.**val1**	число	Параметр предмета 1	Значение параметра
obj.**val2**	число	Параметр предмета 2	Значение параметра
obj.**val3**	число	Параметр предмета 3	Значение параметра
obj.**carried_by**	UID	Кто несет	Персонаж или nil
obj.**worn_by**	UID	На ком одет	Персонаж или nil
obj.**maker**	UID	Возвращает **UID** создателя предмета	UID персонажа или nil
obj.**owner**(uid)	UID	Устанавливает или возвращает **UID** владельца вещи (больше никто не сможет пользоваться)	UID персонажа или nil
char.**varexists**(name)	число	Проверяет у сценария объекта наличие глобальной переменной name.	1 – есть 0 – нет
obj.**g**	строка	о//а/и	Суффикс
obj.**u**	строка	ось/ся/ась/ись	Суффикс
obj.**w**	строка	ое/ый/ая/ые	Суффикс
obj.**q**	строка	ло//ла/ли	Суффикс
obj.**y**	строка	ло//ла/ли	Суффикс
obj.**a**	строка	о//а/ы	Суффикс
obj.**count**	число	Количество предметов в мире (в игре в текущий момент и на ренте)	Количество
obj.**sex**	число	Род предмета	0 – средний 1 – мужской 2 – женский 3 – мн. число
obj.**room**	число	Комната, в которой находится предмет, или тот на ком он одет.	VNUM комнаты
obj.**all** obj.**char** obj.**pc** obj.**npc**	список из UID	Список персонажей в комнате: all – все в комнате char – все PC и чармисы pc – все PC npc – все NPC (мобы не чармисы)	Список персонажей
obj.**put**(id)	-	В зависимости от того, кому принадлежит id, перемещает объект: - персонажу – в инвентарь - комнате – на пол - объекту – внутрь (если контейнер)	-
%obj.uid%	UID	номер предмета, который сохраняется между ребутами (id, возвращаемый тем же calcuid - это только на ребут). Если предмет новый (не был у персонажей), то уид проставляется сразу на месте.	
%obj.skill%	число	Возвращает умение добавляемое этим предметом	

Назначение полей val0 – val4 для разных типов предметов

Тип объекта	Константа ITEM_xxx	VAL0	VAL1	VAL2	VAL3
Источник света	LIGHT			-1 – вечный свет n – сколько часов гореть	
Свиток	SCROLL	Уров. заклин	Закл. 1	Закл. 2	Закл. 3
Палочка	WAND	Уров. заклин	Макс. кол-во	Тек. кол-во	Заклинание
Посох	STAFF	Уров. заклин	Макс. кол-во	Тек. кол-во	Заклинание
Оружие	WEAPON		Кол. бросков	Разм. кости	Тип
Ценности	TREASURE				
Доспех	ARMOR	AC	Броня		
Зелье	POTION	Уров. заклин	Закл. 1	Закл. 2	Закл. 3
Другое	OTHER				
Хлам	TRASH				
Сумка	CONTAINER	Вместимость	Тип ключа	Номер ключа	1 – труп 0 – не труп
Записка	NOTE				
Емкость	DRINKCON	Макс. кол-во	Тек. кол-во	Жидкость	Отравлен
Ключ	KEY				
Еда	FOOD	Насыщение			Отравлен
Деньги	MONEY	Количество			
Ручка	PEN				
Лодка	BOAT				
Колодец	FOUNTAIN	Макс. кол-во	Тек. кол-во	Жидкость	Отравлен
Книга	BOOK		Заклинание		
Магич. Ингр.	INGRADIENT	Интервал применений, уровень	прототип	Кол-во применений	Время последнего применения

Коды типов предметов

1 ..... light source
2 ..... scroll
3 ..... wand
4 ..... staff
5 ..... weapon
8 ..... treasure, not gold
9 ..... armor
10 ..... potion
12 ..... misc object
13 ..... trash
15 ..... container
16..... note
17..... drink container
18..... key
19..... food
20..... money (gold)
21..... pen
22..... boat
23..... fountain
24..... book
25..... magical ingradient

Поля переменных комнат

Операция	Тип	Действие	Результат
room.**name**	строка	Название комнаты	Название
room.**north**	число	Проверка выхода на север	vnum комнаты или nil
room.**east**	число	Проверка выхода на восток	vnum комнаты или nil
room.**south**	число	Проверка выхода на юг	vnum комнаты или nil
room.**west**	число	Проверка выхода на запад	vnum комнаты или nil
room.**up**	число	Проверка выхода вверх	vnum комнаты или nil
room.**down**	число	Проверка выхода вниз	vnum комнаты или nil
room.**vnum**	число	vnum комнаты	vnum комнаты
room.**id**	число	Получение численного значения	Численное значение UID

Операция	Тип	Действие	Результат
		UID комнаты room.	
room.**sectortype**	строка	Тип поверхности в комнате	Гладкий пол Улица … Разбитая дорога
room.**objects**	список из UID	Список объектов на полу комнаты	Список id объектов
room.**people**	UID	Первый персонаж в комнате room.	Персонаж или nil
room.**all** room.**char** room.**pc** room.**npc**	список из UID	Список персонажей в комнате: all – все в комнате char – все PC и чармисы pc – все PC npc – все NPC (мобы не чармисы)	Список персонажей
room.**varexists**(name)	число	Проверяет у сценария объекта комната наличие глобальной переменной name.	1 – есть 0 – нет

Команды DG Script.

Программа на языке DG Script представляет собой последовательность строк. Каждая строка может содержать специальные операторы переменные, знаки арифметических действий и т.д. При интерпретации сценария для каждой строки выполняются три действия: замена переменных, вычисление выражений, выполнение действий.

Все строки, первым не пробельным символом в которых является **‘*’**, интерпретатор рассматривает как комментарий и просто пропускает такие строки.
Командой DG Script является любая стандартная MUD команда или дополнительные операторы DG Script. Имя команды должно быть первым в строке и оставшаяся часть строки рассматривается как параметры команды. Каждая команда должна располагаться на новой строке.

Все команды можно разделить на два типа: управляющие конструкции и функции DG Script.
Управляющие конструкции позволяют создавать ветвления, циклы и т.д. Обработка управляющих конструкций происходит до подстановки переменных.
Функции DG Script представляют собой специальные операторы, которые выполняют функции недоступные командному интерпретатору.
Если команда не была распознана как управляющая конструкция и не похожа на функцию DG Script, то текст передается на обработку командному интерпретатору.

Замена переменных

При замене переменных в строке осуществляется поиск имен переменных, заключенных в кавычки, и их замена на текстовое значение. Осуществляется замена всех переменных в строке. Подробнее о переменных см. в разделе Переменные DG Script.

Вычисление выражений

В процессе выполнения триггера необходимо производить вычисление выражений. В DG Script определены следующие операторы:

```
||.......логическое "или"          >........сравнение >
&&.......логическое "и"            /=.......проверка подстроки
==.......равенство                 -........разность
!=.......неравенство               +........сумма
<=.......сравнение <=              /........частное
>=.......сравнение >=              *........произведение
<........сравнение <               !........логическое отрицание
```

Приоритетов операторов нет, вычисляются по порядку. Для введения приоритетов необходимо использовать скобки.

Управляющие конструкции

Оператор условия

if условие
  операторы
elseif условие
  операторы
else
  операторы
end

Приведенная выше полная форма оператора условия может быть свернута. else/elseif части могут отсутствовать. Важным моментом является то, что для каждого оператора **if** должен быть соответствующий оператор **end**.
При обработке поля *условие* происходит замена переменных и вычисление выражений.

Операторы цикла

while условие
  операторы
done

foreach имя список
  операторы
done

Преждевременный выход из цикла может быть выполнен с использованием оператора **break**.

Цикл while выполняется до тех пор, пока после подстановки переменных и вычисления выражения *условие* будет ненулевым.

Цикл foreach выполняется пока переменная имя не примет все значения из списка. foreach i <список> работает так:
1. Если список пустой - выйти
2. Если триггер не имеет переменной i или значение переменной не равно ни одному элементу списка (разделены пробелами), установить i равной первому элементу и выполнить тело
3. Переменная i равна к-ому элементу списка. Если это последний элемент – выйти, иначе i = след. элемент и выполнить тело.

Особенностью обработки оператора цикла является возможное его долгое выполнение. При выполнении цикла его обработка прерывается каждые 30 проходов и выполнение 100 проходов отражается в системном журнале. В этом случае нужно помнить о том, что результат скрипта будет возвращен после 30 проходов цикла, а не после завершения всего цикла. Т.е. оператор **return** нужно вызывать до вызова длинного цикла, а не после.

Оператор выбора

switch оператор
  case выражение

```
    операторы
  break
  case выражение
    операторы
  break
  default
    операторы
  break
done
```

Оператор семантически похож на оператор языка C (несколько case подряд, значения break, отсутствие default).

Оператор в switch и выражения в case вычисляются при каждом проходе заново.

Примечание: слова **end**, **done** и **case** проверяются (почему-то) без конечных пробелов, поэтому можно писать, например **endif** вместо **end**.

Функции DG Script

Команды не относятся к определенному типу триггера (т.е. могут быть использованы в триггерах моба/объекта/комнаты).

nop

Пустой оператор, ничего не делает.

См. также:

halt

Завершает выполнение сценария. Возвращаемое значение устанавливается оператором **return**.

См. также: return

set *имя текст*

Оператор set присваивает *текст* локальной переменной *имя*. Если такой локальной переменной нет, то создается новая.

См. также: eval, extract, makeuid, calcuid, unset

eval *имя выражение*

Оператор eval вычисляет *выражение* и присваивает вычисленное значение локальной переменной *имя*. Если такой локальной переменной нет, то создается новая.

См. также: set, extract, makeuid, calcuid, unset

extract *имя номер текст*

Оператор extract вычленяет из *текст* слово *номер* (начиная с 1) и присваивает вычлененное значение локальной переменной *имя*. Если такой локальной переменной нет, то создается новая.

См. также: set, eval, makeuid, calcuid, unset

makeuid *имяпеременной id*

Оператор makeiud создает UID-переменную не проверяя наличия объекта с заданным идентификатором. makeuid вычисляет значение выражения *id* создает ссылку на объект с полученным идентификатором и присваивает ссылку локальной переменной *имя*. Если такой локальной переменной нет, то создается новая.

См. также: set, eval, extract, calcuid, unset

calcuid *имяпеременной vnum* (**mob**|**obj**|**room**) [num]

Оператор calcuid создает ссылку на объект с номером *vnum*. calcuid производит поиск объекта заданного типа с заданным *vnum* (vnum выражением быть не может, должен быть задан числом) и создает ссылку на найденный объект. Полученная ссылка присваивается локальной переменной *имя*. Если такой локальной переменной нет, то создается новая.
Если объект найти не удалось, переменная не создается.
Поиск осуществляется следующим образом:

- для комнат – по всему миру
- для объектов - если триггер комнаты: ищем в комнате потом в мире.
  если триггер моба: ищем в инвентаре\эквипе моба потом в комнате где находатся моб, потом в мире.
  если триггер объекта: ищем в инвентаре\эквипе у кого находится объект, потом в комнате, потом в мире. В сумках не ищется.
- для мобов – по всему миру, но не NOWHERE

Если указан num – возвращается найденный объект номер **num** с указанным vnum, в остальных случаях возвращается **первый** найденный объект.
Часто вместо этого оператора удобно использовать поля **world.mob**(vnum), **world.obj**(vnum) и **world.room**(vnum).

См. также: set, eval, extract, makeuid, unset

calcuidall *имяпеременной vnum* (**mob**|**obj**|**room**)

Оператор calcuidall создает список существующих объектов с номером vnum (первые найденные 25шт). calcuidall производит поиск объекта заданного типа с заданным vnum (vnum выражением быть не может, должен быть задан числом) и создает список заданных объектов. Полученная список присваивается локальной текстовой переменной имя. Если такой локальной переменной нет, то создается новая.
Если объект найти не удалось, переменная не создается.
Поиск осуществляется следующим образом:

- для комнат – по всему миру
- для объектов – по всему миру
- для мобов – по всему миру, но не NOWHERE

Во всех случаях возвращается список.

См. также: set, eval, extract, makeuid, unset, calcuid

charuid имяпеременной имяигрока

Оператор charuid аналог caluid для игроков

unset *имяпеременной*

Оператор unset удаляет переменную *имя*. Поиск осуществляется в следуюдем порядке: мир, глобальные, локальные. Удаляется только 1 переменная, даже если есть несколько с разными контекстами.
Мировые и глобальные переменные ищутся с учетом текущего контекста сценария (см. оператор **context**), локальные – без учето оного.

См. также: set, eval, extract, calcuid, makeuid, rdelete

dg_cast '*заклинание***'** *цель*

Оператор dg_cast зачитывает указанное в параметрах заклинание на указанную цель. Заклинание должно быть заключено в символы ‘.

См. также: dg_affect

dg_affect *цель параметр заклинание значение длительность* [*флаг*]

Оператор dg_affect предназначен для установки/снятия того или иного аффекта на персонажа. Если поля параметр и заклинание состоят из нескольких слов, то пробелы должны быть заменены символом ‘_’. Например dg_affect %actor.name% сила подчинить_разум -5 10000.
Цель – персонаж, на которого накладывают аффект. Может быть задан UID.
Параметр – собственно накладываемый аффект. Может быть двух типов: модификатором или аффектом. Модификаторы – это сила, вес, броня и т.д. (см. массив apply_types[], APPLY_xxx). Аффекты – это слепота, яд, ярость и т.д. (см. массив affected_bits[], AFF_xxx).
Заклинание – тип заклинания, наложившего аффект.
Значение – величина измененией выбранного параметра.
Длительность – длительность налагаемого аффекта. Должна быть >=0. Если Длительность =0, то все аффекты типа *заклинание* снимаются с персонажа. Длительность задается тиках или раундах.
Флаг – (необязательный параметр) число битовая маска, управляющая поведением аффекта.
1 – (AF_BATTLEDEC) – длительность уменьшается в бою в раундах
2 – (AF_DEADKEEP) – сохранить аффект после смерти персонажа
4 – (AF_PULSEDEC) – длительность уменьшается вне боя каждый пульс (25.5 раз в секунду)

Элементы массива apply_types[]: сила, ловкость, интеллект, мудрость, телосложение, обаяние, профессия, уровень, возраст, вес, рост, запоминание, макс.жизнь, макс.энергия, деньги, опыт, защита, попадание, повреждение, защита.от.парализующих.заклинаний, защита.от.непонятно.чего, защита.от.окаменяющих.заклинаний, защита.от.вредных.дыханий, защита.от.повреждающей.магии, восст.жизни, восст.энергии, слот.1, слот.2, слот.3, слот.4, слот.5, слот.6, слот.7, слот.8, слот.9, размер, броня, яд, защита.от.боевых.умений, успех.колдовства, мораль, инициатива, религия, поглощение.
Элементы массива affected_bits[]: слепота, невидимость, определение наклонностей, определение невидимости, определение магии, чувствовать жизнь, хождение по воде, освящение, состоит в группе, проклятие, инфравидение, яд, защита от тьмы, защита от света, магический сон, нельзя выследить, привязан, доблесть, подкрадывается, прячется, ярость, зачарован, обездвижен, летит, молчание, настороженность, мигание, верхом или под седлом, не сбежит, свет, освещение, затемнение, определение яда, под мухой, отходняк, декстраплегия, синистроплегия, параплегия, кровотечение, маскировка, дышать водой, торможение, ускорение, защита богов, воздушный щит, огненный щит, ледяной щит, зеркало магии, звезды, каменная рука, призматическая.аура, нанят, силы зла, воздушная аура, огненная аура, ледяная аура

См. также: dg_cast

global *имя*

Оператор делает локальную переменную *имя* глобальной для сценария. Контекст созданной глобальной переменной становится равным текущему контексту сценария (см. оператор **context**). Из списка локальных переменная удаляется.

См. также: worlds, remote

bonus строка

Запускает ивент бонуса, параметры точно такие же как и в команде «бонус» (<двойной|тройной|отменить> [оружейный|опыт|урон] [время])

teleport имя комната [horse followers]

Переносит персонажа по UID или имени в указанную комнату, параметр horse – перенесется с лошадкой, followers – все следуемые за персонажем NPC.

worlds *имя*

Оператор делает локальную переменную *имя* глобальной для мира. Контекст созданной мировой переменной становится равным текущему контексту сценария (см. оператор **context**). Из списка локальных переменная удаляется.

См. также: global, remote

context *число*

Оператор меняет текущий контекст сценария. Новым контекстом будет заданное число. ВНИМАНИЕ, вычисления числа не происходит, только замена переменных.

См. также: global, worlds

remote *имя id*

Оператор создает глобальную переменную *имя* в контексте объекта *id*.
Сначала происходит поиск переменной *имя* в перечне переменных текущего сценария. В локальных переменных поиск осуществляется без учета контекста, в глобальных – с учетом контекста. С использованием *id* происходит поиск объекта-назначения (комната/моб/объект). Для найденного объекта создается глобальная переменная *имя* с текущим контекстом **этого** сценария (см. оператор **context**) и ей присваивается значение найденной переменной (для PC-объектом создаваемый контекст всегда 0). Если у объекта *id* существует глобальная переменная с таким именем и контекстом, то просто изменяется ее значение.

См. также: rdelete

rdelete *имя id*

Оператор удаляет глобальную переменную *имя* в контексте объекта *id*.
С использованием *id* происходит поиск объекта-назначения (комната/моб/объект). Затем происходит поиск переменной *имя* с учетом контекста **этого** сценария в глобальных переменных найденного объекта. Если у объекта *id* существует глобальная переменная с таким именем и контекстом, то она удаляется.

См. также: remote

return число

Оператор изменяет возвращаемое значение сценария. Новым возвращаемым значением сценария становится *число*. При этом выполнение сценария продолжается. При старте сценария возвращаемое значение по умолчанию 1.

См. также: wait, halt, while

wait until *время*
wait *интервал* [(**s**|**t**)]

Оператор приостанавливает выполнение сценария. until-форма позволяет задать абсолютное время возобновления выполнения сценария в виде HH:MM или HHMM. Параметр *интервал* задает величину паузы. Необязательный параметр после него – единицы измерения, а именно: s–секунды, t–тики, нет параметра – пульсы.

См. также:

attach *num id*

Оператор прикрепляет к объекту *id* триггер с номером *num*.

См. также: detach

detach *num id*

Оператор удалает триггер с номером *num* у объекта *id*.

См. также: attach

run *num id*
exec *num id*

Выполняют сценарий *num* объекта *id*. Если сценарии или объект не найдены – выполнение текущего сценария немедленно прекращается. exec выполняет заданный сценарий и возвращается к выполнению текущего. run возвращается к выполнению текущего сценария, только если исполненный сценарий вернул 0.

См. также: attach

otransform vnum

Преобразовывает объект копируя значения (в том числе и таймер) с указанного vnum. Само значение vnum объекта не изменяется.

mtransform vnum

Преобразовывает моба копируя значения с указанного vnum. Само значение vnum не изменяется.

version

Выводит информацию о версии в системный журнал.

См. также:

Примеры триггеров

Пример 1

Задача: раздать всем игрокам на клетке свечки.

Решение:
```
foreach pc %self.pc%
   mload obj 1000
   give свечка %pc.name%
done
```

Пример 2

Задача: Нанести повреждения всей группе нападающего.

Решение:
```
eval gopa %damager.group%
foreach i %gopa%
```

```
   %i.position(6)%
   %i.wait(%random.3%)%
   mdamage %i% %random.100%
done
```

Пример 3

Задача: назначить триггер vnum=1000 объекту vnum=2222

Решение:
```
attach 1000 %world.obj(2222)%
```
или
```
calcuid tmp 2222 obj
attach 1000 %tmp.id%
```

Второй способ предпочтительней в случае, когда несколько триггеров необходимо закрепить за одним объектом. Однако даже в этом случае желательно использовать следующие команды
```
eval tmpid %world.obj(2222)%
attach 1000 %tmpid%
attach 1001 %tmpid%
detach 1002 %tmpid%
```

Пример 4

Задача: создать два списка персонажей, разделенных по половому признаку.

Решение:
```
unset men
unset women
unset other
foreach i %self.pc%
   switch %i.sex%
      case 1
         eval men %men% %i%
      break
      case 2
         eval women %women% %i%
      break
      default
         eval other %other% %i%
      break
   end
done
```

Пример 5

Задача: Обкастовать группу, если лидер делает определенные действия

Решение:
```
eval gr %actor.group%
if %gr.words(1)% == %actor%
   foreach i %gr%
      dg_cast 'свет' %i.name%
   done
end
```

Пример 6

Пример использования цикла общего назначения.

Так пишется цикл стандартными средствами

```
eval firstchar %self.people%
eval num 0
while %firstchar% && (%num% < 5)
```

```
   eval pc %firstchar.next_in_room%
   if %firstchar.vnum% == -1
      mteleport %firstchar% 25610
      eval num %num%+1
   end
   if %pc%
      makeuid firstchar %pc.id%
   else
      set firstchar 0
   end
done
```

```
eval num 5
foreach i %pcs%
   mteleport %i% 25610
   eval num %num%-1
   if %num% == 0
      break
   end
done
```

Обратите внимание, что если не считать количество обработанных персонажей (переменная num), то тело цикла сократится до 1 (!) строки.

Так пишется цикл с использованием foreach

```
eval pcs %self.pc%
```

Примечание: на самом деле, опытный билдер никогда не напишет то, что представлно слева (цикл стандартными средствами). Первое, что бросается в глаза – команду

```
      makeuid firstchar %pc.id%
```

можно (и нужно) заменить на команду

```
      eval firstchar %pc%
```

Кроме того, опытный билдер поймет, что конструкция

```
   if %pc%
      makeuid firstchar %pc.id%
   else
      set firstchar 0
   end
```

в данном контексте эквивалентна единственной (!!!) команде

```
   eval firstchar %pc%
```

Пример 7

Использование команды wat/mat.

Зачастую в триггере необходимо перенести персонаж в другое место мира и выдать соответсвующие сообщения в старом и новом месте. Часто для этого используют такой код:

```
wechoaround %actor% %actor.name% исчез%actor.q%.
wat 27400 wechoaround %actor% %actor.name% появил%actor.u%.
wteleport %actor% 27400 horse
```

Этот код неправильный!!! Несмотря на то, что во второй строке действие происходит на клетке 27400, команда wechoaround реализована так, что если ее параметром является UID (как в этом случае), то в качестве целевой клетки для сообщения используется месторасположение объекта UID. Т.е. в приведенном коде оба сообщения будут выданы в начальной точке расположения персонажа, после чего персонаж будет перемещен.

Исправить код можно двумя способами:

1. Изменить цель wechoaround, например

```
wat 27400 wechoaround %actor.name% %actor.name% появил%actor.u%.
```

2. Поменять местами 2 и третью строку примера. В этом случае, команду wat можно вообще не использовать.

Пример 8

Использование в сообщениях суффиксов и модификаторов.

Часто в триггере необходимо вывести сообщение, но суффиксы и окончания будут выглядеть по разному в зависимости от пола персонажа, о котором пойдет речь. Эту проблему можно решить так:

```
if %actor.sex%==1
   mechoaround %actor% %actor.name% залез в воздушный шар.
   mechoaround %actor% %actor.name% улетел на воздушном шаре.
else
   mechoaround %actor% %actor.name% залезла в воздушный шар.
   mechoaround %actor% %actor.name% улетела на воздушном шаре.
end
```

Видно, что ничем, кроме окончаний данные тексты не отличаются. В таких случаях намного эффективнее использовать поля переменной char, которые описывают окончания и суффиксы согласо пола персонажа. Для приведенного примера это будет выглядеть так:

```
mechoaround %actor% %actor.name% залез%actor.q% в воздушный шар.
mechoaround %actor% %actor.name% улетел%actor.g% на воздушном шаре.
```

Но это еще не все! Если вокруг стоят персонажи, которые не видят %actor% им сообщение должно быть выдано в соответствующем виде, где имя %actor% заменено на “кто-то”. Для этого можно использовать специальный модификатор, а именно ‘~’. Т.о. окончательно код триггера будет выглядеть

```
mechoaround %actor% ~%actor.name% залез%actor.q% в воздушный шар.
mechoaround %actor% ~%actor.name% улетел%actor.g% на воздушном шаре.
```

Примечание относительно ‘~’. Т.к. символ ‘~’ является служебным в файлах зоны, для указания символа ‘~’ в тексте триггера необходимо его указывать дважды! Т.е., например, ~~%actor.name%. При загрузке и интерпретации триггера этот двойной символ ‘~~’ будет преобразоватьв одинарный.

Все возможные суффиксы можно посмотреть в таблице “Поля переменных моба”.
Все возможные модификаторы приведены в таблице ниже.

Символ модификатора	Значение			
	Обеъкт не найден	Объект – цель сообщения	Объект виден	Объект не виден
~	“Кто-то”	“Вы”	Имя, именительный	”Кто-то”
@	“чей-то”	“Ваш”	Имя, родительный	”кого-то”
^	“чей-то”	“Ваш”	“его/ее/их/его”	“чей-то”
&	“Он”	“Вы”	“он/она/они/оно”	“Он”
*	“ему”	“Вам”	“ему/ей/им/ему”	“ему”
`	“что-то”	–	Имя, именительный	“что-то”

Пример 9

Использование списка списков. Задача есть несколько списков и список имен этих списков. Нужно поэлементно просмотреть все списки и создать общий список всех элементов.

```
* Пример приводится для 3х списков.
* Задаю три списка ‘свои’, ‘чужие’ и ‘все’.
* Список ‘test’ содержит имена изначальных списков.
set свои 1 2 3 4 5
set чужие 10 11 12 13
set все n q r s А Б В Г
set test свои чужие все

* К сожалению, нужно сделать переменные глобальными для триггера,
* иначе не работает косвенная подстановка имени переменной .global()
global свои
global чужие
global все

unset r
* Вложенный цикл соберет полную строку в переменной ‘r’
foreach i %test%
  foreach j %self.global(%i%)%
    eval r %r% %j%
  done
done
%echo% %r%
* Резльтатом будет строка
* 1 2 3 4 5 10 11 12 13 n q r s А Б В Г
```

Пример 10

Здесь попробую разъяснить использование боевых триггеров мобов.

(разъяснение будет позже)

Пример 11

Продемонстрирую возможности триггеров на относительн большой задаче. Ниже приведены 2 триггера, которые позволяют мобу играть с игроком в известную логическую игру. Моб загадывает комбинацию из заданных образцов, после чего игрок пытается угадать ее. При каждой попытке моб сообщает сколько в предлагаемой игроком комбинации элементов совпадают по виду(цвету) и сколько по месторасположению.

Чтобы не увеличивать размер триггера, комментарии будут даваться справа.
Первым идет SPEECH триггер загабывания комбинации

```
#4050                                   номер триггера
Согласие на игру~                       название
0 d0 0
да согласен давай~                      ключевые слова
set N 4                                 количество элементов в загаданной
                                        мобом комбинации, можно при желании
                                        изменить
set образцы a b c d e f g h             образцы, из которых выбираются
                                        элементы для комбинации (могут быть
                                        словами, цифрами и т.д.)
set попытка 0                           счетчик попыток

дум                                     моб задумывается (загадывает
                                        комбинацию)

set i 1
while %i% <= %N%                        случайный выбор элементов
   eval комбинация %комбинация% %random.num(%образцы.words%)%
   eval i %i%+1
done

global попытка                          перевод в глобальные переменные,
global образцы                          чтобы были доступны из другого
global комбинация                       триггера

set i 1
while %i% <= %образцы.words%
   unset комбинация_расчет              сброс предыдущих расчетов
   eval i %i%+1
done

set i 1
while %i% <= %N%
   context %комбинация.words(%i%)%
   eval комбинация_расчет %комбинация_расчет% + 1
   global комбинация_расчет             массив 'комбинация_расчет' содержит
   eval i %i%+1                         кол-во каждого образца в загаданной
done                                    комбинации

context 0

wait 2 s
г Готово. Отгадывай, %actor.name%!      загадал, можно отгадывать

~
```

Теперь COMMAND триггер – попытка угадать

```
#4051
Попытка угадать~
0 c0 0
комбинация~                             Ключевое слово. Триггер вызывается,
                                        например 'комбинация a f g g'
```

```
1   if %arg.words% != %комбинация.words%
2      г Неправильное количество элементов в комбинации.
3      г Попробуй еще раз.
4      halt
5   end
6
7   set i 1
8   while %i% <= %arg.words%                Цикл по всем аргументам команды
9      set j 1
10     while %j% <= %образцы.words%         Поиск аргумента в образцах
11        if %arg.words(%i%)% == %образцы.words(%j%)%
12           break
13        end
14        eval j %j%+1
15     done
16     if %j% > %образцы.words%
17        г Что-то странное ты предлагаешь. Что такое '%arg.words(%i%)%'?
18        г Попробуй еще раз.
19        halt
20     end
21     eval try %try% %j%                   Преобразование аргументов в номера
22                                          образцов
23     eval i %i%+1
24  done
25
26  set i 1
27  while %i% <= %образцы.words%
28     unset try_расчет                     Сброс предыдущего расчета
29     eval i %i%+1
30  done
31
32  set i 1
33  while %i% <= %try.words%
34     context %try.words(%i%)%
35     eval try_расчет %try_расчет% + 1
36     global try_расчет                    Создание массива счетчиков образцов
37     eval i %i%+1
38  done
39
40  set color 0
41  set place 0
42
43  set i 1
44  while %i% <= %образцы.words%            Подсчет угаданных видов образцов
45     context %i%
46     eval j %комбинация_расчет%
47     if %try_расчет% < %j%
48        eval j %try_расчет%
49     end
50     eval color %color%+%j%
51     eval i %i%+1
52  done
53
54  set i 1
55  while %i% <= %try.words%                Подсчет угаданных мест образцов
56     if %try.words(%i%)% == %комбинация.words(%i%)%
57        eval color %color%-1              Если угадано место, то счетчик
58        eval place %place%+1              угаданных цветов уменьшить.
59     end
60     eval i %i%+1
61  done
62
63  set i 1
64  while %i% <= %try.words%                Восстановление аргументов
```

```
   eval ответ %ответ% %образцы.words(%try.words(%i%)%)%
   eval i %i%+1
done
eval ответ %ответ% : %place%, %color%   Подготовка ответа
eval попытка %попытка%+1
context %попытка%                       Сохранение ответа с контекстом
global ответ                            текущей попытки (история)
context 0
global попытка                          Сохранение нового значения попытки

set i 1
set j %попытка%
while %i% <= %j%                        Вывод истории ответов во всех
   context %i%                          попытках
   г %ответ%
   eval i %i%+1
done

if %color% == %try.words%
   г Молодец, угадал%actor.g%.          Все! Угадал все места.
end
context 0
~

```