Скриптовые функции

ExecConsoleCommand

ExecConsoleCommand — выполнить консольную команду

Синтаксис

`ExecConsoleCommand`(input-string);

Описание

Исполняет входную строку *`input-string`* так, как если бы данная строка была введена пользователем в игровой консоли.

`input-string` передается на исполнение как есть и может содержать все что угодно, включая команды со специальными маркерами '`@`' (способ вызова скриптовых команд), '`#`' и '`$`' (способы вызова иных специальных команд).

Исполняемая *`input-string`* отображается в консоли:

```
> ExecConsoleCommand("gfx_noshadows")
gfx_noshadows
gfx_noshadows string='0' float='0'
```

GetGameVar

GetGameVar — Возвращает значение общеигровой переменной

Синтаксис

`GetGameVar`(name);

Описание

Функция возвращает значение общеигровой переменной с именем *`name`* если она есть и пустую строку если ее нет.

`name` — имя переменной

---

length

length — определяет длину массива(таблицы)

Synopsis

`length`(array);

Описание

Определим понятие "массив" как специальный случай таблицы, в которой ключами являются только числа, причем нумерация идет от нуля и без пропущенных элементов. Функция *`length`* определяет количество элементов в этом массиве..

`array` — массив

GetDifficulty

GetDifficulty — определяет уровень сложности игры

Синтаксис

`GetDifficulty`(`void`);

Описание

Функция возвращает уровень сложности игры. Определены следующие константы для идентификации уровней сложности: `DIFFICULTY_NORMAL`, `DIFFICULTY_HARD`, `DIFFICULTY_HEROIC`. Константы упорядочены по возрастанию, так что можно использовать конструкции вида `GetDifficulty() > DIFFICULTY_NORMAL` и т. д.

Load

Load — загружает игру из указанного файла

Синтаксис

`Load`(fileName);

Описание

Загружает игру из указанного файла

`fileName` — имя указанного в свойствах карты (*`resources->saveFilenames`*) текстового файла в котором находится локализуемое имя файла сохранения игры

---

mod

mod — Возвращает остаток от деления одного числа на другое

Синтаксис

mod(x, y);

Описание

Функция возвращает остаток от деления своего первого аргумента на второй. Оба аргумента должны быть числами (не обязательно целыми). Второй аргумент не должен быть равен нулю.

---

postEvent

postEvent — послать сообщение

Синтаксис

postEvent(event-name, arg1 = -1, arg2 = -1);

Описание

Посылает сообщение, задаваемое *event-name*, через общий механизм рассылки сообщений.

event-name — должен быть строкой с именем сообщения, например:

```
postEvent("Combat_Retreat")
```

С помощью данной команды можно эмулировать, например, подачу команд с клавиатуры.

---

print

print — вывести текстовое представление аргументов в консоль

Синтаксис

print(...);

Описание

Выводит текстовое представление аргумента(-ов) в консоль.

`print` работает с любыми типами параметров:

- числа, строки и `nil` выводятся литерально
- функции и userdata-объекты заменяются на маркеры типа "`<function>`"
- для таблиц выводится их содержимое

Пример:

```
> print(2, " sdf ", print)
2 sdf <C-function>
```

Save

Save — сохраняет игру в указанный файл

Синтаксис

`Save`(fileName);

Описание

Сохраняет игру в указанный файл.

`fileName` — имя указанного в свойствах карты (*`resources->saveFilenames`*) текстового файла в котором находится локализуемое имя файла сохранения игры

SetGameVar

SetGameVar — Устанавливает значение общеигровой переменной

Синтаксис

`SetGameVar`(name, value);

Описание

Функция меняет значение общеигровой переменной с именем *`name`* на *`value`*. Если такой переменной нет, она будет создана.

`name` — имя переменной

`value` — значение, в которое нужно установить переменную

---

sleep

sleep — приостановить на время текущий поток исполнения

Синтаксис

sleep(number-of-segments);

Описание

Останавливает работу текущего потока исполнения на указанное время. Время задается в игровых сегментах.

Значение параметра *number-of-segments* должно быть ≥ 1.

Команда `sleep` необходима для создания скриптов, которые должны выполнять периодические действия в течение долгого времени (например, скрипты, реализующие сценарии карт):

```
while 1 do
    sleep(100)
    print("another 100 game segments has passed")
end
```

Важный эффект команды `sleep` в том, что в результате ее выполнения управление передается от текущего скриптового потока обратно планировщику потоков скриптовой машины, которая таким образом получает возможность дать поработать другим потокам исполнения и, в конце концов, возвратить управление обратно игре.

Например, если приведенный выше скрипт запустить на исполнение, убрав из него вызов `sleep(100)`, то игра "зависнет" – поток, в котором будет исполняться скрипт, никогда не вернет управление скриптовой машине, а та – не вернет управление игре.

---

sqrt

sqrt — Возвращает квадратный корень числа

Синтаксис

sqrt(x);

Описание

Функция возвращает квадратный корень из своего единственного аргумента, который должен быть неотрицательным числом.

---

startThread

startThread — запустить новый поток исполнения

Синтаксис

`startThread`(func);

Описание

Запускает указанную функцию в новом потоке исполнения.

Параметр *`func`* должен быть скриптовой функцией — *`func`* не может быть, например, числом или одной из описываемых в данном руководстве команд (так как они реализованы на C).

```
local immortal = "immortal-archer";

function eternalLoop()
    while 1 do
        if not exist(%immortal) then
            resurrect(%immortal)
        end
        sleep(1)
    end
end

print("Starting resurrection loop")
startThread(eternalLoop)
```

AddHeroCreatures

AddHeroCreatures — добавить отряд существ в армию героя

Синтаксис

`AddHeroCreatures`(heroname, creatureID, quantity);

Описание

Добавляет *`quantity`* существ типа *`creatureID`* в армию героя с именем *`heroname`*.

`heroname` — это внутреннее имя героя, задаваемое при создании карты.

Команда работает независимо от того, где герой находится: герой может отсутствовать на карте (быть в городе или среди убитых (?)).

Боевые механизмы (баллиста, катапульта и т.п.) добавлять нельзя, можно только существа.

AddObjectCreatures

AddObjectCreatures — добавить отряд существ в армию объекта

Синтаксис

`AddObjectCreatures`(objectName, creatureID, quantity);

Описание

Добавляет *`quantity`* существ типа *`creatureID`* в армию объекта с именем *`objectName`*. Добавлять существ можно любому объекту карты, который может иметь армию (в том числе монстрам и героям). Монстрам можно добавлять только существ того же типа, что и содержащиеся в монстре.

`objectName` — имя объекта.

`creatureID` — тип существ

`quantity` — количество существ

Боевые механизмы (баллиста, катапульта и т.п.) добавлять нельзя, можно только существа.

BlockGame

BlockGame — блокирует интерфейс пользователя и работу AI

Синтаксис

`BlockGame`(void);

Описание

Команда блокирует интерфейс пользователя, работу AI и управление камерой. Таким образом, мир останавливается и единственным источником игровых событий остается скрипт. Также команда останавливает бегущих героев (с сохранением пути). Для разблокирования игры нужно использовать команду *`UnblockGame`*. Блокировки складываются, то есть игра будет разблокирована, когда число вызовов *`UnblockGame`* будет равно числу вызовов *`BlockGame`*.

CalcAverageMonstersTier

CalcAverageMonstersTier — вычисляет средний уровень существ, стоящих на карте.

Синтаксис

CalcAverageMonstersTier(floor = -1);

Описание

Функция вычисляет средний уровень существ на заданном уровне карты, либо, если уровень не задан — на всех уровнях сразу. При подсчете не учитывается количество монстров в группе, т. е. все группы при подсчете равноценны.

floor — номер уровня, на котором производится подсчет, -1 для подсчета на всех уровнях (задано по умолчанию).

CalcHeroMoveCost

CalcHeroMoveCost — вычисляет стоимость передвижения героя в заданную точку

Синтаксис

CalcHeroMoveCost(heroName, x, y, floorID = -1);

Описание

Функция вычисляет стоимость передвижения героя в заданную точку (единица измерения — movepoint). Если не указан номер уровня, используется тот, на котором стоит герой. Функцию можно вызывать только для героев принадлежащих компьютерным игрокам. При формировании пути учитываются только статичные объекты. Если заданная точка недоступна, возвращается -1.

heroName — имя героя

x, y — координаты тайла

floorID — номер уровня (по умолчанию = -1, что означает "уровень на котором стоит герой")

CanMoveHero

CanMoveHero — определяет, возможно ли передвижение героя в заданную точку

Синтаксис

CanMoveHero(heroName, x, y, floorID = -1);

Описание

Функция возвращает `true`, если заданная точка достижима для героя. В противном случае возвращается `false`. Если не указан номер уровня, используется тот, на котором стоит герой. Функцию можно вызывать только для героев принадлежащих компьютерным игрокам.

`heroName` — имя героя

`x, y` — координаты тайла

`floorID` — номер уровня (по умолчанию = `-1`, что означает "уровень на котором стоит герой")

ChangeHeroStat

ChangeHeroStat — меняет параметры героя

Синтаксис

`ChangeHeroStat`(heroName, statID, delta);

Описание

Функция изменяет значение указанного параметра героя на величину *`delta`*.

Менять можно следующие параметры: experience, attack, defence, spell power, knowledge, luck, morale, move points, mana points.

`delta` может принимать положительные (параметр увеличивается) и отрицательные (параметр уменьшается) значения, для всех параметров, кроме experience, так как experience может только расти, для него *`delta`* может быть только положительным числом или нулем.

Собственно значения всех параметров не могут быть отрицательными. Значения параметров move points, mana points дополнительно ограничены сверху величинами move points max и mana points max соответственно. Если в результате изменения значение параметра выходит за допустимые границы, оно обрезается.

`heroName` — имя героя

`statID` — идентификатор параметра, может принимать следующие значения:

`STAT_EXPERIENCE` — опыт героя

`STAT_ATTACK` — уровень атаки

`STAT_DEFENCE` — уровень защиты

`STAT_SPELL_POWER` — сила заклинаний

`STAT_KNOWLEDGE` — начитанность

`STAT_LUCK` — удача

`STAT_MORALE` — мораль

`STAT_MOVE_POINTS` — количество оставшихся очков хода

`STAT_MANA_POINTS` — текущее количество очков маны

`delta` — значение, на которое нужно изменить параметр

CreateArtifact

CreateArtifact — создает артефакт на карте

Синтаксис

`CreateArtifact`(artifactName, artifactType, x, y, floorID);

Описание

Создает в клетке *`( x, y )`* на уровне *`floorID`* артефакт типа *`artifactType`* и задает ему имя *`artifactName`*. Если клетка на которую нужно поставить артефакт занята (на ней стоит другой объект или есть объект, для которого она является интерактивной), артефакт ставится на одну из ближайших свободных клеток.

`artifactName` — имя артефакта, которое в дальнейшем можно использовать в других скриптовых командах.

`artifactType` — тип артефакта.

`x, y` — координаты клетки, на которую нужно поставить артефакт.

`floorID` — номер уровня, на который нужно поставить артефакт.

CreateMonster

CreateMonster — создает монстра на карте

Синтаксис

`CreateMonster`(*monsterName*,

creatureType,
creaturesCount,
x,
y,
floorID,
mood= MONSTER_MOOD_AGGRESSIVE,
courage= MONSTER_COURAGE_CAN_FLEE_JOIN,
rotation= 0);

Описание

Создает в клетке *(x, y)* на уровне *floorID* монстра с *creaturesCount* существ типа *monsterType* и задает ему имя *monsterName*. Если клетка на которую нужно поставить монстра занята (на ней стоит другой объект или есть объект, для которого она является интерактивной), монстр ставится на одну из ближайших свободных клеток. Последние два параметра влияют на выбор поведения монстров при встрече с героем (в соответствии со стандартной логикой этого взаимодействия).

monsterName — имя монстра, которое в дальнейшем можно использовать в других скриптовых командах.

creatureType — тип существ, составляющих монстра.

creaturesCount — количество существ, составляющих монстра

x, y — координаты клетки, на которую нужно поставить монстра.

floorID — номер уровня, на который нужно поставить монстра.

mood, courage — эти параметры влияют на выбор поведения монстров при встрече с героем, могут принимать следующие значения:

MONSTER_MOOD_FRIENDLY,

MONSTER_MOOD_AGGRESSIVE,

MONSTER_MOOD_HOSTILE,

MONSTER_MOOD_WILD и

MONSTER_COURAGE_ALWAYS_JOIN,

MONSTER_COURAGE_ALWAYS_FIGHT,

MONSTER_COURAGE_CAN_FLEE_JOIN соответственно.

rotation — угол поворота монстра (в градусах, по умолчанию = 0).

Пред.

DenyAIHeroFlee

DenyAIHeroFlee — выключает передвижение камеры при передвижении героев

Синтаксис

`DenyAIHeroFlee`(heroName, isDenied, enemyHeroName = "");

Описание

Запрещает герою AI сбегать из боя.

Важно понимать, что если разрешить сбегать (в *`isDenied`* передать *`nil`*), то это разрешение может быть перекрыто другими запретами.

Если указать третий параметр, то запрет будет установлен/снят только для определенного противника.

Если третий параметр — пустая строка (она такая по умолчанию), то запрет ставиться/снимается для любых противников, но информация о других запретах остается. Таким образом, если снять запрет на сбегание из любых боев, то другие запреты останутся.

`heroName` — имя героя для которого ставится или снимается запрет.

`isDenied` — значение 1 запрещает сбегать из боев, значение 0 — удаляет запрет (а не разрешает сбегать).

`enemyHeroName` — имя героя противника, для боя с которым ставится или снимается запрет на сбегание.

Пред.

DenyAIHeroesFlee

DenyAIHeroesFlee — выключает передвижение камеры при передвижении героев

Синтаксис

`DenyAIHeroesFlee`(playerID, isDenied, enemyHeroName = "");

Описание

Запрещает всем героям AI сбегать из боя.

Если указать третий параметр, то запрет будет установлен/снят только для определенного противника.

Если третий параметр — пустая строка (она такая по-умолчанию), то запрет ставиться/снимается для любых противников, но информация о других запретах остается. Таким образом, если снять запрет на сбегание из любых боев, то другие запреты останутся.

playerID – игрок, для героев которого ставится или снимается запрет.

isDenied — значение 1 запрещает сбегать из боев, значение 0 — удаляет запрет (а не разрешает сбегать).

enemyHeroName — имя героя противника, для боя с которым ставится или снимается запрет на сбегание.

Пред.

SetAIHeroFleeControl

SetAIHeroFleeControl — выключает передвижение камеры при передвижении героев

Синтаксис

SetAIHeroFleeControl(heroName, isUnique);

Описание

Позволяет выключать следование общим правилам запрета сбегания для героев AI

heroName — герой, которому включается/выключается следование общим запретам.

isUnique – значение 1 выключает общие запреты сбегания из боев (установленные через DenyAIHeroesFlee), значение 0 — включает обратно.

DeployReserveHero

DeployReserveHero — ставит на карту героя, находящегося в резерве у игрока

Синтаксис

DeployReserveHero(heroName, x, y, floor);

Описание

При создании карты некоторые герои могут быть зарезервированы за определенными игроками. Такие герои не будут появляться в тавернах и не могут наниматься игроками обычным способом. Чтобы поставить зарезервированного за игроком героя на карту,

нужно использовать команду *DeployReserveHero*, передав ей в качестве параметров имя героя и местоположение на карте, в которое его нужно установить. Если клетка на карте на которую должен быть поставлен герой по каким-либо причинам недоступна, герой будет поставлен на одну из ближайших к ней клеток. Убитые, сбежавшие и пропавшие из списка принадлежащих игроку героев любым другим способом зарезервированные герои помещаются обратно в резерв и могут быть вновь поставлены на карту той же командой. При этом у них восстанавливаются армии, заданные в редакторе. Для удаления героя из резерва игрока, чтобы он мог быть доступен в тавернах остальных игроков необходимо использовать функцию *UnreserveHero*.

heroName — имя зарезервированного за игроком героя.

x, y — координаты клетки карты, на которую нужно поставить героя.

floor — номер уровня, на который нужно поставить героя.

Пред.

DisableCameraFollowHeroes

DisableCameraFollowHeroes — выключает передвижение камеры при передвижении героев

Синтаксис

DisableCameraFollowHeroes (disableOwn, disableAlly, disableEnemy) ;

Описание

В обычном режиме игры, когда герой игрока, союзника или врага движется, камера центрируется на этом герое и движется вместе с ним . Данная скриптовая команда позволяет выключить слежение камеры за героями игрока, союзника или врага, т.е. герой будет двигаться, но камера будет оставаться на прежнем месте (но не будет блокироваться, т.е. можно будет управлять ею стрелками, мышкой, а также из скрипта, например, командой MoveCamera).

disableOwn — значение 1 выключает движение камеры за героями игрока, значение 0 — включает обратно.

disableAlly — значение 1 выключает движение камеры за героями союзников, значение 0 — включает обратно.

disableEnemy — значение 1 выключает движение камеры за героями врагов, значение 0 — включает обратно.

EnableAIHeroHiring

EnableAIHeroHiring — включает/выключает покупку AI героев в таверне города.

Синтаксис

`EnableAIHeroHiring`(playerID, townName, enable);

Описание

Функция позволять запрещать/разрешать AI покупать героев в таверне указанного города (изначально покупка разрешена).

`playerID` — идентификатор игрока

`townName` — имя города

`enable` — `true` для включения, `false` для выключения

EnableHeroAI

EnableHeroAI — включает/выключает контроль AI над заданным героем

Синтаксис

`EnableHeroAI`(heroName, enable);

Описание

Функция позволять включать и выключать контроль AI над заданным героем. Функцию можно использовать только в одиночной игре. При вызове функции для героя, управляемого игроком — человеком генерируется ошибка. Если управление героем находится в том же состоянии, в которое его пытается установить функция, не производится никаких действий.

`heroName` — имя героя

`enable` — `true` для включения управления AI, `false` для выключения

IsObjectExists

IsObjectExists — определяет, существует ли данный объект на карте

Синтаксис

```
IsObjectExists(objectName);
```

Описание

Возвращает `true`, если объект с заданным именем существует на карте и `false`, если нет. Эту функцию можно использовать, задавая в качестве имени объекта `OBJECT_GRAIL`.

`objectName` — имя объекта

GenerateMonsters

GenerateMonsters — генерирует монстров на карте

Синтаксис

```
GenerateMonsters(,
                 ,
                 ,
                 ,
                 );
```

Описание

Функция генерирует монстров в случайных местах на игровой карте. Создается от *`countGroupsMin`* до *`countGroupsMax`* групп существ с количеством существ в каждой группе от *`countInGroupMin`* до *`countInGroupMax`*.

`monsterTypeID` — идентификатор типа существ

`countGroupsMin` — минимальное количество групп существ (должно быть >= 0)

`countGroupsMax` — максимальное количество групп существ (должно быть >= *`countGroupsMin`*)

`countInGroupMin` — минимальное количество существ в группах (должно быть > 0)

`countInGroupMax` — максимальное количество существ в группах (должно быть >= *`countInGroupMin`*)

GetCurrentPlayer

GetCurrentPlayer — определяет текущего игрока

Синтаксис

GetCurrentPlayer(void);

Описание

Функция возвращает идентификатор игрока, который ходит в данный момент.

GetDate

GetDate — возвращает текущее игровое время (день, неделю, месяц или день недели)

Синтаксис

GetDate(dateTypeID);

Описание

Возвращает текущее игровое время. Параметр задает, что нужно определить: текущий день, неделю, месяц или день недели. По-умолчанию (при вызове без параметров) возвращает текущий день.

dateTypeID — тип информации о текущем времени, которую нужно определить

Возможные значения:

DAY — день

WEEK — неделя

MONTH — месяц

DAY_OF_WEEK — день недели

ABSOLUTE_DAY — тоже, что и *DAY*

GetHeroCreatures

GetHeroCreatures — определяет количество существ заданного вида, находящихся под командованием героя

Синтаксис

GetHeroCreatures(heroName, creatureID);

Описание

Возвращает количество существ заданного вида, находящихся под командованием указанного героя.

heroName — имя героя

creatureID — идентификатор типа существа

GetHeroLevel

GetHeroLevel — определить уровень героя

Синтаксис

GetHeroLevel(heroname);

Описание

Возвращает текущий уровень героя *heroname*.

Ошибка

- если героя с именем *heroname* нет (IsHeroAlive(*heroname*) возвращает `nil`/`false`)

GetHeroSkillMastery

GetHeroSkillMastery — возвращает уровень владения героя навыком

Синтаксис

GetHeroSkillMastery(heroName, skillID);

Описание

Функция возвращает для заданного героя уровень владения заданным навыком (0 — не знает, 1 — basic, 2 — advanced, 3 — expert, 4 — extra expert).

Навык/уровень может быть как изученный, так и данный герою артефактом.

heroName — имя героя

skillID — идентификатор навыка

GetHeroStat

GetHeroStat — определяет параметры героя

Синтаксис

GetHeroStat(heroName, statID);

Описание

Функция возвращает значение указанного параметра героя. Определять можно следующие параметры: experience, attack, defence, spell power, knowledge, luck, morale, move points, mana points. Значения параметров определяются с учетом всех эффектов (артефактов и т. д.).

heroName — имя героя

statID — идентификатор параметра, может принимать следующие значения:

STAT_EXPERIENCE — опыт героя

STAT_ATTACK — уровень атаки

STAT_DEFENCE — уровень защиты

STAT_SPELL_POWER — сила заклинаний

STAT_KNOWLEDGE — начитанность

STAT_LUCK — удача

STAT_MORALE — мораль

STAT_MOVE_POINTS — количество оставшихся очков хода

STAT_MANA_POINTS — текущее количество очков маны.

GetHeroTown

GetHeroTown — определяет город, в котором находится герой

Синтаксис

`GetHeroTown`(heroName);

Описание

Возвращает имя города, в гарнизоне которого находится герой или `nil` если герой находится на карте.

`heroName` — имя героя

GetObjectCreatures

GetObjectCreatures — определяет количество существ заданного вида, содержащихся в армии объекта

Синтаксис

`GetObjectCreature`(objectName, creatureID);

Описание

Возвращает количество существ заданного вида, находящихся в армии указанного объекта. Команду можно применять к любым объектам карты, которые имеют армию (в том числе к монстрам и героям).

`objectName` — имя объекта

`creatureID` — идентификатор типа существа

GetObjectiveProgress

GetObjectiveProgress — определяет прогресс выполнения задания

Синтаксис

`GetObjectiveProgress`(objectiveName, playerID = PLAYER_1);

Описание

Функция возвращает прогресс выполнения задания с именем *`objectiveName`*. Для заданий общих для всех игроков параметр *`playerID`* задает игрока, для которого нужно определить

прогресс выполнения им задания (если параметр не задан, определяется прогресс выполнения задания первым игроком). Для заданий специфичных для игрока параметр *playerID* игнорируется.

Количество шагов выполнения заданий управляемых вручную определяется количеством заданных в редакторе комментариев прогресса выполнения задания. Задания "захватить заданные объекты на карте" и "уничтожить заданные нейтральные армии" имеют количество шагов выполнения, равное количеству объектов карты, которые нужно захватить и количеству армий, которые нужно уничтожить соответственно. Остальные задания не имеют шагов прогресса выполнения и могут быть только либо выполненными, либо нет.

objectiveName — имя задания

playerID — идентификатор игрока, для которого нужно определить прогресс выполнения им задания (для принадлежащих конкретному игроку заданий игнорируется, по умолчанию равен `PLAYER_1`)

GetObjectiveState

GetObjectiveState — определяет состояние задания

Синтаксис

```
GetObjectiveState(objectiveName, playerID = PLAYER_1);
```

Описание

Функция возвращает состояние задания с именем *objectiveName* для определенного игрока. Для заданий принадлежащих конкретному игроку параметр *playerID* игнорируется. Для общих заданий, если параметр *playerID* задан, он указывает для какого игрока нужно определить состояние задания, в противном случае — возвращается состояние задания для первого игрока.

objectiveName — имя задания

playerID — идентификатор игрока, для которого нужно определить состояние задания (для заданий, принадлежащих конкретному игроку, параметр игнорируется, по умолчанию равен `PLAYER_1`).

Список возможных состояний заданий:

`OBJECTIVE_SCENARIO_INFO` — специальное состояние, которое указывает на то, что данное задание на самом деле является описанием сценария текущей карты.

`OBJECTIVE_UNKNOWN` — задание неизвестно игроку. Если для задачи установлен флаг видимости (см. функции *`GetObjectiveState`* / *`SetObjectiveState`*) в интерфейсе заданий отображается туманное описание задания.

`OBJECTIVE_ACTIVE` — задание в данный момент выполняется игроком.

`OBJECTIVE_COMPLETED` — задание выполнено.

`OBJECTIVE_FAILED` — задание провалено.

GetObjectOwner

GetObjectOwner — определяет принадлежность находящихся на карте объектов

Синтаксис

`GetObjectOwner`(objectName);

Описание

Возвращает идентификатор игрока, которому принадлежит указанный объект. Если объект никому не принадлежит — возвращается PLAYER_NONE.

`objectName` — имя объекта

GetObjectsInRegion

GetObjectsInRegion — определяет, какие объекты находятся в заданной области

Синтаксис

`GetObjectsInRegion`(regionName, objectType);

Описание

Функция возвращает имена стоящих на карте объектов заданного типа, находящихся внутри региона. Объекты, не имеющие имени, игнорируются.

`regionName` — имя региона

`objectType` — тип объектов, нахождение которых внутри региона нужно проверить

На данный момент команда работает только для одного типа объектов:

`OBJECT_HERO` — герои

GetObjectPosition

GetObjectPosition — определяет положение объекта на карте

Синтаксис

`GetObjectPosition`(objectName);

Описание

Функция возвращает три значения — *`X`* и *`Y`* координаты тайла, на котором стоит объект (центрального, если их несколько), и номер уровня, на котором находится объект. Эту функцию можно использовать в качестве имени объекта `OBJECT_GRAIL`.

`objectName` — имя объекта

GetPlayerHeroes

GetPlayerHeroes — возвращает имена героев принадлежащих игроку

Синтаксис

`GetPlayerHeroes`(playerID);

Описание

Функция возвращает массив, содержащий имена героев, принадлежащих заданному игроку.

`playerID` — номер игрока.

GetPlayerResource

GetPlayerResource — получить количество ресурсов у игрока

Синтаксис

`GetPlayerResource`(player, resourceKind);

Описание

Возвращает, сколько у игрока *`player`* ресурсов вида *`resourceKind`*.

`player` — номер игрока от `1` до `8`. Вместо чисел можно использовать глобальные константы PLAYER_1 до PLAYER_8.

`resourceKind` — число от `0` до `6`, либо одна из констант `WOOD`, `ORE`, `MERCURY`, `CRYSTAL`, `SULFUR`, `GEM`, `GOLD`.

Команда работает для активных и проигравших игроков. Запрос ресурсов у игрока, не принимающего участия в текущей игре, является ошибкой.

Ошибка

- если аргументы не прошли проверку на допустимость значений
- если указанный игрок в игре не участвует

GetPlayerState

GetPlayerState — получить текущее состояние игрока

Синтаксис

`GetPlayerState`(player);

Описание

Возвращает состояние участия/победы/поражения игрока *`player`*: 0 — игрок в игре не участвует, 1 — игрок еще играет, 2 — игрок проиграл, 3 — ни то, ни другое, ни третье (возможно, победил, но нельзя сказать определенно).

`player` — номер игрока от `1` до `8`. Вместо чисел можно использовать глобальные константы PLAYER_1 до PLAYER_8.

GetStandState

GetStandState — определяет текущее состояния "стенда" на карте

Синтаксис

GetStandState(standName);

Описание

Функция возвращает номер текущего состояния "стенда" с заданным именем.

standName — имя стенда

GetStandStatesCount

GetStandStatesCount — определяет количество возможных состояний "стенда" на карте

Синтаксис

GetStandStatesCount(standName);

Описание

Функция возвращает количество возможных состояний "стенда" с заданным именем. Состояния нумеруются подряд от нуля.

standName — имя стенда

GetTownBuildingLevel

GetTownBuildingLevel — определяет уровень постройки здания в городе

Синтаксис

GetTownBuildingLevel(townName, buildingID);

Описание

Функция возвращает уровень заданного здания в городе. Уровень 0 означает, что здание не было построено.

townName — имя города

buildingD — идентификатор типа строения, может принимать следующие значения:

TOWN_BUILDING_TOWN_HALL,

```
TOWN_BUILDING_FORT,

TOWN_BUILDING_MARKETPLACE,

TOWN_BUILDING_SHIPYARD,

TOWN_BUILDING_TAVERN,

TOWN_BUILDING_BLACKSMITH,

TOWN_BUILDING_MAGIC_GUILD,

TOWN_BUILDING_DWELLING_1,

TOWN_BUILDING_DWELLING_2,

TOWN_BUILDING_DWELLING_3,

TOWN_BUILDING_DWELLING_4,

TOWN_BUILDING_DWELLING_5,

TOWN_BUILDING_DWELLING_6,

TOWN_BUILDING_DWELLING_7,

TOWN_BUILDING_GRAIL,

TOWN_BUILDING_WONDER,

TOWN_BUILDING_HAVEN_TRAINING_GROUNDS,

TOWN_BUILDING_HAVEN_MONUMENT_TO_FALLEN_HEROES,

TOWN_BUILDING_HAVEN_HOSPITAL,

TOWN_BUILDING_HAVEN_STABLE,

TOWN_BUILDING_HAVEN_FARMS,

TOWN_BUILDING_INFERNO_INFERNAL_LOOM,

TOWN_BUILDING_INFERNO_ORDER_OF_FIRE,

TOWN_BUILDING_INFERNO_HALLS_OF_HORROR,

TOWN_BUILDING_INFERNO_SACRIFICIAL_PIT,

TOWN_BUILDING_DUNGEON_ALTAR_OF_ELEMENTS,

TOWN_BUILDING_DUNGEON_RITUAL_PIT,
```

TOWN_BUILDING_DUNGEON_TRADE_GUILD,

TOWN_BUILDING_DUNGEON_TREASURE_DIG_SITE,

TOWN_BUILDING_DUNGEON_HALL_OF_INTRIGUE,

TOWN_BUILDING_ACADEMY_LIBRARY,

TOWN_BUILDING_ACADEMY_ARCANE_FORGE,

TOWN_BUILDING_ACADEMY_ARTIFACT_MERCHANT,

TOWN_BUILDING_ACADEMY_TREASURE_CAVE,

TOWN_BUILDING_ACADEMY_ELEMENTAL_ENCLAVE,

TOWN_BUILDING_PRESERVE_AVENGERS_BROTHERHOOD,

TOWN_BUILDING_PRESERVE_MYSTIC_POND,

TOWN_BUILDING_PRESERVE_SPARKLING_FONTAINS,

TOWN_BUILDING_PRESERVE_BLOOMING_GROVE,

TOWN_BUILDING_PRESERVE_TREANT_SAMPLING,

TOWN_BUILDING_NECROMANCY_AMPLIFIER,

TOWN_BUILDING_NECROMANCY_UNHOLY_TEMPLE,

TOWN_BUILDING_NECROMANCY_UNEARHED_GRAVES,

TOWN_BUILDING_NECROMANCY_DRAGON_TOMBSTONE,

TOWN_BUILDING_NECROMANCY_SHROUD_OF_DARKNESS.

Внимание: указание специфичного для города здания для города другой расы может привести к ошибочным результатам.

GetTownBuildingLimitLevel

GetTownBuildingLimitLevel — определяет ограничение на уровень постройки здания в городе

Синтаксис

GetTownBuildingLimitLevel(townName, buildingID);

Описание

Функция возвращает ограничение на возможный уровень заданного здания в городе. Уровень 0 означает, что здание не может быть построено.

`townName` — имя города

`buildingD` — идентификатор типа строения, список возможных значений находится в описании команды *`GetTownBuildingLevel`*.

GetTownBuildingMaxLevel

GetTownBuildingMaxLevel — определяет максимальный уровень постройки здания в городе

Синтаксис

`GetTownBuildingMaxLevel`(townName, buildingID);

Описание

Функция возвращает наивысший возможный уровень заданного здания в городе. Уровень 0 означает, что здание не может быть построено.

`townName` — имя города

`buildingD` — идентификатор типа строения, список возможных значений находится в описании команды *`GetTownBuildingLevel`*.

GetTownHero

GetTownHero — возвращает героя, находящегося в гарнизоне города.

Синтаксис

`GetTownHero`(townName);

Описание

Возвращает имя героя находящегося в гарнизоне указанного города или `nil` если в гарнизоне города никого нет.

townName — имя города

GiveArtefact

GiveArtefact — дать герою артефакт

Синтаксис

GiveArtefact(*heroname*, *artefactID*, *[bindToHero = 0]*);

Описание

Дает герою *heroname* артефакт *artefactID*.

artefactID — номер артефакта от 0 до 55 или символьная константа из списка

bindToHero — привязывать ли артефакт к герою (сделать его непередаваемым)

Ошибка

- если героя с именем *heroname* нет (IsHeroAlive(*heroname*) возвращает `nil`/`false`)

GiveBorderguardKey

GiveBorderguardKey — дает игроку ключ определенного цвета

Синтаксис

GiveBorderguardKey(player, color);

Описание

Дает игроку ключ заданного цвета. Если у игрока такой ключ уже есть — не производит никаких действий.

player — номер игрока.

color — цвет ключа, может принимать следующие значения:

`RED_KEY`, `BLUE_KEY`, `GREEN_KEY`, `YELLOW_KEY`, `ORANGE_KEY`, `TEAL_KEY`, `PURPLE_KEY`, `TAN_KEY`.

LevelUpHero

LevelUpHero — дает герою столько очков опыта, сколько необходимо для получения следующего уровня

Синтаксис

`LevelUpHero`(heroName);

Описание

Дает герою *`heroName`* очки опыта в количестве, достаточном для получения следующего уровня. Функция возвращает `true` если герою удалось дать уровень и `nil` если это невозможно (герой уже имеет максимальный возможный уровень).

`heroName` — имя героя

GiveHeroSkill

GiveHeroSkill — дает герою заданный базовый навык

Синтаксис

`GiveHeroSkill`(heroName, skillID);

Описание

Функция делает попытку дать герою заданный базовый навык. Если это невозможно (например, герой уже имеет mastery "`EXPERT`" для этого навыка, или для него нет места на линейке), функция возвращает `false`, в противном случае возвращается `true`.

`heroName` — имя героя

`skillID` — идентификатор навыка

GiveHeroWarMachine

GiveHeroWarMachine — дает герою машину заданного типа

Синтаксис

`GiveHeroWarMachine` (heroName, warMachineType) ;

Описание

Функция пытается дать герою указанную машину. Если у героя она уже есть, функция возвращает `nil`, в противном случае функция возвращает `not nil`.

`heroName` — имя героя

`warMachineID` — идентификатор типа машины

HasArtefact

HasArtefact — есть ли у героя артефакт

Синтаксис

`HasArtefact` (heroname, artefactID) ;

Описание

Возвращает, есть ли у героя *`heroname`* артефакт *`artefactID`*.

`artefactID` — номер артефакта от 0 до 54 *(в описании артефактов на сайте H5 написано: «всего артефактов 54, один может быть раздвоится, будет 55», — однако в игре их только 53)*.

Надо еще (надо ли?) определить символьные константы для артефактов, чтобы употреблять `HasArtefact("Agrael", SWORD_OF_RUINS)` вместо `HasArtefact("Agrael", 0)`.

Ошибка

- если героя с именем *`heroname`* нет (IsHeroAlive(*`heroname`*) возвращает `nil`/`false`)

HasBorderguardKey

HasBorderguardKey — определяет, есть ли у игрока ключ определенного цвета

Синтаксис

`HasBorderguardKey` (player, color) ;

Описание

Возвращает `not nil` если у игрока есть ключ заданного цвета и `nil` если нет.

`player` — номер игрока.

`color` — цвет ключа, может принимать следующие значения:

`RED_KEY`, `BLUE_KEY`, `GREEN_KEY`, `YELLOW_KEY`, `ORANGE_KEY`, `TEAL_KEY`, `PURPLE_KEY`, `TAN_KEY`.

HasHeroSkill

HasHeroSkill — проверяет наличие навыка у героя

Синтаксис

`HasHeroSkill`(heroName, skillID);

Описание

Функция возвращает `true`, если герой имеет заданный навык (вне зависимости от того, имеет его сам герой, или этот навык дается ему благодаря артефакту), в противном случае возвращается `false`.

`heroName` — имя героя

`skillID` — идентификатор навыка

HasHeroWarMachine

HasHeroWarMachine — определяет, есть ли у героя машина заданного типа

Синтаксис

`HasHeroWarMachine`(heroName, warMachineType);

Описание

Возвращает `not nil`, если у героя есть машина указанного типа и `nil` в противном случае.

`heroName` — имя героя

warMachineID — идентификатор типа машины

IsHeroAlive

IsHeroAlive — жив ли герой?

Синтаксис

IsHeroAlive(heroname);

Описание

Возвращает, существует ли герой *heroname*, и если существует, жив ли он.

Живым считается герой, принадлежащий какому-либо активному игроку (для которого GetPlayerState() возвращает true). В момент запроса герой может находиться где угодно: на поверхности, в подземелье, в городе, на корабле.

heroname — это внутреннее скриптовое имя героя (не только то, что задается при создании карты).

IsHeroLootable

IsHeroLootable — определяет, можно ли отобрать у героя артефакты, победив его

Синтаксис

IsHeroLootable(heroName);

Описание

Возвращает `not nil`, если артефакты героя в случае победы над ним передаются победителю и `nil`, если они остаются у побежденного героя.

heroName — имя героя

IsObjectEnabled

IsObjectEnabled — определяет, взаимодействует ли интерактивный объект с героем стандартным образом

Синтаксис

`IsObjectEnabled`(objectName);

Описание

Если функция возвращает `true`, интерактивный объект при заходе в него героя ведет себя стандартным образом. Если функция возвращает значение `false`, при заходе героя в этот объект не происходит ничего кроме вызова функции обработчика триггера, если она установлен. По умолчанию интерактивный объект ведет себя стандартным образом.

`objectName` — имя интерактивного объекта

IsObjectInRegion

IsObjectInRegion — определяет, находится ли объект в заданной области

Синтаксис

`IsObjectInRegion`(objectName, regionName);

Описание

Функция возвращает `true`, если объект находится в заданном регионе и `false`, если нет.

Примечание: функция корректно работает только для однотайловых объектов.

`objectName` — имя объекта

`regionName` — имя региона

IsObjectiveVisible

IsObjectiveVisible — определяет, отображается ли задание в интерфейсе конкретного игрока.

Синтаксис

`IsObjectiveVisible`(objectiveName, playerID = PLAYER_1);

Описание

Функция возвращает `true`, если задание отображается в интерфейсе игрока и `false`, если нет. Параметр *`playerID`* игнорируется для заданий, принадлежащих конкретному игроку. Для общих заданий, если параметр *`playerID`* задан, то он определяет игрока для которого нужно определить видимость задания, если нет — определяется видимо ли задание первому игроку.

`objectiveName` — имя задания

`playerID` — идентификатор игрока (для заданий, принадлежащих конкретному игроку, параметр игнорируется, по умолчанию равен `PLAYER_1`).

IsObjectVisible

IsObjectVisible — определяет, виден ли объект игроку

Синтаксис

`IsObjectVisible`(playerID, objectName);

Описание

Функция возвращает `true`, если заданный объект виден игроку, и `false` – если нет.

Примечание: функция корректно работает только для однотайловых объектов.

`playerID` — идентификатор игрока

`objectName` — имя объекта

IsRegionBlocked

IsRegionBlocked — определяет, заблокирован ли регион для прохождения героями определенного игрока

Синтаксис

`IsRegionBlocked`(regionName, playerID);

Описание

Функция возвращает `not nil`, если регион заблокирован для прохождения героями заданного игрока и `nil` если нет.

`regionName` — имя региона

`playerID` — идентификатор игрока

KillMobs

KillMobs — удаляет монстров с карты

Синтаксис

`KillMobs`(monsterTypeID);

Описание

Функция удаляет монстров заданного типа с игровой карты.

`monsterTypeID` — идентификатор типа монстра

KnowHeroSpell

KnowHeroSpell — проверяет знание заклинания героем

Синтаксис

`KnowHeroSpell`(heroName, spell);

Описание

Функция возвращает `true`, если герой знает заданное заклинание (вне зависимости от того, знает его сам герой, или это заклинание дается ему благодаря артефакту), в противном случае возвращается `false`.

`heroName` — имя героя

`spell` — идентификатор заклинания

Список идентификаторов заклинаний находится в common.lua.

Loose

Loose — при вызове функции игрок-человек проигрывает

Синтаксис

```
Loose(void);
```

Описание

Данная функция предназначена для использования только в режиме одиночной игры. При вызове в сетевом режиме генерируется ошибка. После вызова функции игрок-человек немедленно проигрывает.

MarkObjectAsVisited

MarkObjectAsVisited — помечает выключенный интерактивный объект как использованный данным героем

Синтаксис

```
MarkObjectAsVisited(objectName, heroName);
```

Описание

Команда должна быть вызвана при взаимодействии героя с выключенным командой *SetObjectEnabled* интерактивным объектом.

objectName — имя интерактивного объекта

heroName — имя героя, который взаимодействует с объектом

MessageBox

MessageBox — выводит на экран сообщение

Синтаксис

```
MessageBox(messageName, callback = "");
```

Описание

Функция выводит на экран сообщение в виде диалога с единственной кнопкой "OK". Параметр *callback* (если он указан) задает имя функции, которая будет вызвана после нажатия на "OK".

messageName — имя сообщения в базе ресурсов

callback — имя функции, которая будет вызвана после нажатия на "OK"

MoveCamera

MoveCamera — перемещает камеру в заданную точку карты

Синтаксис

MoveCamera(x, y, floorID, zoom = 50, pitch = pi/2, yaw = 0, noZoom = 0, noRotate = 0);

Описание

Перемещает камеру в заданную позицию и устанавливает ей заданные значения приближения и угла наклона.

x, y, floorID — эти параметры задают клетку на карте, на которую будет смотреть камера

zoom — параметр определяет насколько далеко будет расположена камера от заданной клетки

pitch — угол наклона камеры (`0` — камера смотрит вдоль горизонта, `pi/2` — вертикально вниз)

yaw — угол поворота камеры (0 — север)

noZoom — установить в 1, чтобы камера не меняла zoom (приближение) при движении

noRotate — установить в 1, чтобы камера не вращалась при движении

MoveHero

MoveHero — отдает герою приказ двигаться в определенную точку.

Синтаксис

MoveHero(heroName, x, y, floorID = -1);

Описание

Функция отдает герою приказ двигаться в заданную точку. Если не указан номер уровня, предполагается, что герой должен пройти в точку с заданными координатами на своем уровне. Функцию можно вызывать только для героев принадлежащих компьютерным игрокам и выключенным AI. Если заданная точка недоступна, функция генерирует ошибку. Если у героя в пути заканчиваются movepoint'ы, он продолжает движение на следующем ходу.

heroName — имя героя

x, y — координаты тайла

floorID — номер уровня (по умолчанию = -1, что означает "уровень на котором стоит герой")

MoveHeroRealTime

MoveHeroRealTime — заставляет героя двигаться в определенную точку.

Синтаксис

MoveHeroRealTime(heroName, x, y, floorID = -1);

Описание

Функция отдает герою приказ двигаться в заданную точку. Если не указан номер уровня, предполагается, что герой должен пройти в точку с заданными координатами на своем уровне. Если заданная точка недоступна, функция генерирует ошибку. Движение начинается сразу после вызова команды, вне зависимости от того, чей в данный момент ход. Скрипт продолжает работу сразу после выполнения команды, не дожидаясь, когда герой достигнет точки назначения. Во время движения блокируется пользовательский интерфейс и действия AI. Если у героя в пути заканчиваются очки движения или он встречает препятствие — он останавливается, не взаимодействуя с объектом (в частности это означает, что герой не способен пользоваться телепортами, лодками и т.д.).

heroName — имя героя

x, y — координаты тайла

floorID — номер уровня (по умолчанию = -1, что означает "уровень на котором стоит герой")

GetAllNames

GetAllNames — выдать список имен объектов в игре (на карте)

Синтаксис

`GetAllNames`(filter = 0);

Описание

Возвращает строку со списком имен объектов в игре (на карте). Имена существ разделяются пробелом.

Параметр *`filter`* задает вид объектов, чьи имена должны быть включены в результат. Возможные значения *`filter`*:

- `0` — имена героев активных игроков

OpenCircleFog

OpenCircleFog — открывает туман войны в заданном круге.

Синтаксис

`OpenCircleFog`(x, y, floorID, range, playerID);

Описание

Функция открывает туман войны игрока на уровне *`floorID`* в круге с центром *`(x, y)`* и радиусом *`range`*.

`player` — идентификатор игрока

`x, y` — координаты центра круга

`floorID` — номер уровня

`range` — радиус круга

OpenPuzzleMap

OpenPuzzleMap — "виртуально" посещает заданное количество обелисков для данного игрока

Синтаксис

`OpenPuzzleMap`(player, numObelisks);

Описание

"Виртуально" посещает заданное количество обелисков для данного игрока. При этом реальные обелиски должны находиться где-нибудь на карте т.к. именно из их количества игра вычисляет сколько кусочков Puzzle Map открывается при каждом посещении. Эффективное число обелисков — 36 — если их больше, то только первые 36 реальных или виртуальных посещений будут открывать по кусочку Puzzle Map, а остальные — только показывать уже полностью открытую.

`player` — идентификатор игрока

`numObelisks` — количество "виртуально" посещаемых обелисков

OpenRegionFog

OpenRegionFog — открывает туман войны в заданном регионе.

Синтаксис

`OpenRegionFog`(player, regionName);

Описание

Функция открывает туман войны игрока в заданном регионе.

`player` — идентификатор игрока

`regionName` — имя региона

Play2DSound

Play2DSound — проигрывает 2D звук

Синтаксис

`Play2DSound`(soundName);

Описание

Проигрывает 2D звук с заданным именем.

Если звук определен в базе как циклический, функция возвращает число — идентификатор звука, который затем можно передать функции *`StopPlaySound`* , чтобы остановить проигрывание звука.

Если звук определен как не циклический, возвращается `-1`.

`soundName` — имя звука

Play3DSound

Play3DSound — проигрывает 3D звук

Синтаксис

`Play3DSound`(soundName, x, y, floor);

Описание

Проигрывает 3D звук с заданным именем. Источником звука является тайл с координатами *`x`* и *`y`*, находящийся на уровне *`floor`*.

Если звук определен в базе как циклический, функция возвращает число — идентификатор звука, который затем можно передать функции *`StopPlaySound`* , чтобы остановить проигрывание звука.

Если звук определен как не циклический, возвращается `-1`.

`soundName` — имя звука

PlayObjectAnimation

PlayObjectAnimation — проигрывает анимацию на объекте карты

Синтаксис

`PlayObjectAnimation`(objectName, animName, action);

Описание

Проигрывает на указанном объекте карты анимацию *animName*.

action определяет, как именно анимация будет играться. В игре выделены следующие виды действий:

- INVISIBLE — (особое) объект становится невидимым
- IDLE — анимация играется циклически до тех пор, пока не будет заменена какой-либо другой анимацией
- ONESHOT_STILL — анимация проигрывается однократно, по окончании объект остается в позе последнего кадра анимации
- ONESHOT — анимация проигрывается однократно, по окончании для объекта запускается его стандартная idle-анимация
- MOVE — анимация проигрывается циклически по пути в пространстве, по окончании – стандартная idle-анимация
- NON_ESSENTIAL — то же что ONESHOT, но анимация играется только в том случае, если объект не "занят", то есть если на объекте не играется ничего, кроме IDLE-анимации

animName — символьное имя анимации ("idle00", "attack00", "hit" и т.п.), его следует знать заранее. Если у объекта анимации с именем *animName* нет, то команда ничего не делает.

Действия INVISIBLE и MOVE данной командой не поддерживается.

PlayVisualEffect

PlayVisualEffect — проигрывает заданный визуальный эффект (без звука) на карте

Синтаксис

PlayVisualEffect(effectName, objectName="", tagName="", x=0,y=0,z=0, rot=0, floor=0);

Описание

Проигрывает визуальный эффект с заданным именем. Центром эффекта является позиция заданного объекта *objectName* к которой прибавляется смещение *x,y,z*, а результирующий угол определяется углом объекта, к которому прибавляется *rot*. Если объект не задан (пустая строка), то эффект играется относительно системы координат карты (тайла с координатами *x,y*, высота берется из ландшафта и уже к ней прибавляется *z*), а этаж задается параметром *floor*.

Если эффект определен в базе как циклический, то для того, чтобы была возможность остановить эффект(ы), нужно задать параметр *tagName* (имя-тэг). Останавливает эффекты функция *StopVisualEffects*.

effectName — ссылка на ресурс эффекта

objectName — имя объекта относительно которого будет играться эффект (можно не указывать)

tagName — имя для управления эффектом, необязательный параметр — по умолчанию имя пустое

x,y,z — координаты в тайлах относительно объекта или карты (если объект не указан)

rot — угол поворота относительно объекта или карты (если объект не указан)

floor — этаж (используется если объект не указан), 0 — поверхность, 1 — подземелье

QuestionBox

QuestionBox — выводит на экран сообщение, требующее подтверждения

Синтаксис

QuestionBox(messageName, callbackYes, callbackNo);

Описание

Функция выводит на экран сообщение в виде диалога с кнопками "OK" и "Cancel". Параметр *callbackYes* задает имя функции, которая будет вызвана если игрок нажмет кнопку "OK". Параметр *callbackNo* задает имя функции, которая будет вызвана если игрок нажмет кнопку "Cancel".

messageName — имя сообщения в базе ресурсов

callbackYes — имя функции, которая будет вызвана после нажатия на "OK"

callbackNo — имя функции, которая будет вызвана после нажатия на "Cancel"

---

random

random — Возвращает случайное число

Синтаксис

random(top);

Описание

Функция возвращает случайное целое число в диапазоне от нуля до `top − 1`. Аргумент функции должен быть положительным целым числом.

RazeBuilding

RazeBuilding — разрушает объект на карте

Синтаксис

`RazeBuilding`(objectName);

Описание

Убирает с карты заданное строение (города тоже) и ставит на его место статичный объект (разрушенная версия объекта). Объект "разрушенный" определяется полем *`RazedStatic`* в *`AdvMap*Shared`* (где вместо * — одно из названий — Building, Dwelling, Town и др.). Если это поле пусто — объект невозможно разрушить. Набор блокирующих клеток разрушенного объекта должен совпадать с соответствующим набором исходного объекта. Разрушенный объект, как и любой другой статичный объект не должен иметь интерактивных клеток.

`objectName` — имя объекта, который нужно разрушить

RazeTown

RazeTown — разрушает город на карте

Синтаксис

`RazeTown`(townName);

Описание

Убирает с карты заданный город и ставит на его место статичный объект — разрушенный город. Объект "разрушенный город" определяется полем *`razed`* в *`AdvMapTownShared`*. Если это поле пусто — город невозможно разрушить. Набор блокирующих клеток разрушенного города должен совпадать с соответствующим набором исходного города. Разрушенный город, как и любой другой статичный объект не должен иметь интерактивных клеток.

`townName` — имя города

RegionToPoint

RegionToPoint — определяет координаты и номер уровня точечного региона

Синтаксис

`RegionToPoint`(regionName);

Описание

Функция возвращает координаты *`X`* и *`Y`* и номер уровня, на котором находится вырожденный регион, состоящий из единственного тайла. Если функции передается невырожденный регион, генерируется ошибка.

`regionName` — имя региона

RemoveArtefact

RemoveArtefact — забирает артефакт у героя

Синтаксис

`RemoveArtefact`(*heroname*, *artefactID*);

Описание

Забирает у героя *`heroname`* артефакт *`artefactID`*.

`artefactID` — номер артефакта от 0 до 55 или символьная константа из списка

Ошибка

- если героя с именем *`heroname`* нет (IsHeroAlive(*`heroname`*) возвращает `nil`/`false`)
- если у героя нет ни одного подобного артефакта (*`advmap.HasArtefact(heroName,artefactID)`* возвращает `nil`/`false`)

RemoveHeroCreatures

RemoveHeroCreatures — убирает существ из армии героя

Синтаксис

`RemoveHeroCreatures`(heroname, creatureID, quantity);

Описание

Убирает *`quantity`* существ типа *`creatureID`* в из армии героя с именем *`heroname`*. Если в армии героя существ заданного типа меньше, чем задано в параметре *`quantity`*, команда удаляет всех существ заданного типа из армии героя. Если кроме удаляемых существ в армии героя никого нет, команда удаляет всех существ кроме одного.

`heroname` — имя героя.

Команда работает и для не активных героев (находящихся в тавернах, тюрьмах и т.д.).

Боевые механизмы (баллиста, катапульта и т.п.) удалять нельзя, можно только существа.

`creatureID` — тип существа

`quantity` — количество

RemoveHeroWarMachine

RemoveHeroWarMachine — отнимает у героя машину заданного типа

Синтаксис

`RemoveHeroWarMachine`(heroName, warMachineType);

Описание

Функция пытается отнять у героя указанную машину. Если у героя ее нет или отнять ее невозможно, функция возвращает `nil`, в противном случае функция возвращает `not nil`.

`heroName` — имя героя

`warMachineID` — идентификатор типа машины

Примечание: катапульта у героя есть всегда и отнять ее невозможно

RemoveObject

RemoveObject — удаляет объект с игровой карты

Синтаксис

`RemoveObject`(objectName);

Описание

Удаляет объект с заданным именем с игровой карты.

`objectName` — имя объекта

RemoveObjectCreatures

RemoveObjectCreatures — убирает существ из армии объекта

Синтаксис

`RemoveObjectCreatures`(objectName, creatureID, quantity);

Описание

Убирает *`quantity`* существ типа *`creatureID`* в из армии объекта с именем *`objectName`*. Если в армии объекта существ заданного типа меньше, чем задано в параметре *`quantity`*, команда удаляет всех существ заданного типа из армии объекта. Команду можно применять ко всем объектам карты, которые имеют армии (в том числе к монстрам и героям). Если объект — это герой и кроме удаляемых существ в армии героя никого нет, команда удаляет всех существ кроме одного. Если объект — это монстр и удаляются все составляющие его существа, он снимается с карты.

`objectName` — имя объекта

`creatureID` — тип существа

`quantity` — количество

Боевые механизмы (баллиста, катапульта и т.п.) удалять нельзя, можно только существа.

ResetHeroCombatScript

ResetHeroCombatScript — Сбрасывает скрипт, который был бы запущен при бое с данным героем

Синтаксис

ResetHeroCombatScript(heroName);

Описание

Указывает, что при бое с данным героем не должен быть запущен скрипт. Если бой происходит в месте, для которого задан другой боевой скрипт (город, гарнизон и т.д.), будет запущен этот скрипт.

heroName — имя героя

ResetObjectFlashlight

ResetObjectFlashlight — убирает с объекта карты точечный источник света

Синтаксис

ResetObjectFlashlight(objectName);

Описание

Команда удаляет с выбранного объекта карты дополнительный источник света, добавленный командой *SetObjectFlashlight*.

objectName — имя объекта.

SetAIHeroAttractor

SetAIHeroAttractor — Устанавливает цель для заданного AI героя

Синтаксис

SetAIHeroAttractor(objectName, heroName, priority);

Описание

Функция модифицирует оценку домика для заданного AI героя.

Примечание: убрать модификацию оценки домика можно, задав приоритет 0.

Примечание: функция корректно работает только для неподвижных объектов.

objectName — имя объекта

heroName — имя героя

priority — приоритет цели -1..2.

-1 — Оценка домика сильно уменьшается.

0 — Оценка домика не меняется.

1 — Оценка домика сильно увеличивается. Но AI не будет рисковать героями пади этого домика.

2 — Оценка домика равной победе в партии. AI будет игнорировать все опасности ради захвата домика.

SetAIPlayerAttractor

SetAIPlayerAttractor — Устанавливает цель для всех героев AI игрока.

Синтаксис

SetAIPlayerAttractor(objectName, playerID, priority);

Описание

Функция модифицирует оценку домика для всех героев заданного AI игрока.

Примечание: убрать модификацию оценки домика можно задав приоритет 0.

Примечание: функция корректно работает только для неподвижных объектов.

objectName — имя объекта

playerID — идентификатор игрока

priority — приоритет цели -1..2.

-1 — Оценка домика сильно уменьшается.

0 — Оценка домика не меняется.

1 — Оценка домика сильно увеличивается. Но AI не будет рисковать героями пади этого домика.

2 — Оценка домика равной победе в партии. AI будет игнорировать все опасности ради захвата домика.

SetCombatLight

SetCombatLight — меняет освещение на всех боевых аренах карты

Синтаксис

`SetCombatLight`(lightName);

Описание

Функция подменяет источник рассеянного света на всех боевых аренах карты на указанный в параметре команды.

`lightName` — указатель ресурса источника рассеянного света в ресурсной базе

SetHeroCombatScript

SetHeroCombatScript — Указывает скрипт, который будет запущен при бое с данным героем

Синтаксис

`SetHeroCombatScript`(heroName, scriptName);

Описание

Функция позволяет указать скрипт, который будет запущен при сражении с данным героем. Скрипт будет запущен только если герой будет находиться под управлением компьютерного игрока и для места где происходит бой (город, гарнизон и т.д.) не задан другой скрипт. Скрипт автоматически сбрасывается, когда игрок теряет этого героя любым способом.

`heroName` — имя героя

`scriptName` — указатель на скрипт

SetHeroLootable

SetHeroLootable — задает, можно ли отобрать у героя артефакты, победив его

Синтаксис

SetHeroLootable(heroName, enable);

Описание

Параметр *enable* определяет, будут ли артефакты героя в случае победы над ним передаваться победителю (not nil) или же будут оставаться у побежденного героя (nil).

heroName — имя героя

enable — параметр определяет, отбираются ли у героя артефакты при поражении в бою

SetAmbientLight

SetAmbientLight — меняет освещение на заданном уровне карты

Синтаксис

SetAmbientLight(floorID, lightName, fade = false, *time* = 1);

Описание

Устанавливает новый источник рассеянного света на заданном уровне карты. Список доступных источников задается в описании карты.

floorID — номер уровня, на котором нужно сменить освещение

lightName — имя источника рассеянного света

fade — параметр определяет должно ли изменение освещенности уровня происходить постепенно (*fade* = true) или мгновенно (*fade* = false) (старый и новый источники света должны различаться только цветовыми компонентами)

time — время изменения

SetObjectEnabled

SetObjectEnabled — позволяет включать и выключать взаимодействие интерактивного объекта с героем стандартным образом

Синтаксис

`SetObjectEnabled`(objectName, enable);

Описание

Команда позволяет включать/выключать стандартное поведение интерактивного объекта при заходе в него героя. По умолчанию объект ведет себя стандартным образом. При вызове данной команды с параметром *`enable`* равным `false` при заходе героя в этот объект не происходит ничего кроме вызова функции обработчика триггера, если она установлен.

`objectName` — имя интерактивного объекта

`enable` — флаг определяющий будет ли объект при заходе в него героя вести себя стандартным образом

SetObjectiveProgress

SetObjectiveProgress — задает прогресс выполнения задания

Синтаксис

`SetObjectiveProgress`(objectiveName, step, playerID = PLAYER_1);

Описание

Функция задает прогресс выполнения задания с именем *`objectiveName`*. Для заданий общих для всех игроков параметр *`playerID`* задает игрока, для которого нужно установить степень выполнения задания (если параметр не задан, устанавливается степень выполненности задания первым игроком). Для заданий специфичных для игрока параметр *`playerID`* игнорируется.

Управлять степенью выполненности можно только для заданий, управляемых вручную. Количество шагов выполнения заданий управляемых вручную определяется количеством заданных в редакторе комментариев прогресса выполнения задания.

`objectiveName` — имя задания

`step` — шаг выполнения задания

`playerID` — идентификатор игрока, для которого нужно определить прогресс выполнения им задания (для принадлежащих конкретному игроку заданий игнорируется, по умолчанию равен `PLAYER_1`)

SetObjectiveState

SetObjectiveState — меняет состояние задания

Синтаксис

`SetObjectiveState`(objectiveName, state, playerID = PLAYER_1);

Описание

Функция изменяет состояние задания с именем *`objectiveName`* для определенного игрока. Для заданий принадлежащих конкретному игроку параметр *`playerID`* игнорируется. Для общих заданий, если параметр *`playerID`* задан, он указывает для какого игрока нужно изменить состояние задания, в противном случае — меняется состояние задания для первого игрока.

`objectiveName` — имя задания

`state` — состояние, в которое нужно установить задание

`playerID` — идентификатор игрока, для которого меняется состояние задания (для заданий, принадлежащих конкретному игроку, параметр игнорируется, по умолчанию равен `PLAYER_1`).

Таблица допустимых переходов между состояниями:

`OBJECTIVE_SCENARIO_INFO` -> нет (задание на самом деле является описанием сценария)

`OBJECTIVE_UNKNOWN` -> `OBJECTIVE_ACTIVE` (если активация задания не зависит от выполнения других заданий) и `OBJECTIVE_FAILED` (если задание определено как управляемое вручную).

`OBJECTIVE_ACTIVE` -> `OBJECTIVE_COMPLETED`, `OBJECTIVE_FAILED` (если задание определено как управляемое вручную).

`OBJECTIVE_COMPLETED` -> `OBJECTIVE_ACTIVE` и `OBJECTIVE_FAILED` (если для задания указано, что оно может потерять статус "выполнено" и оно определено как управляемое вручную).

`OBJECTIVE_FAILED` -> нет

Примечание: при установке состояния задания в `OBJECTIVE_ACTIVE` оно автоматически становится видимым игроку (см. функции *`IsObjectiveVisible`* / *`SetObjectiveVisible`*).

SetObjectiveVisible

SetObjectiveVisible — отображает / скрывает задание в интерфейсе конкретного игрока.

Синтаксис

`SetObjectiveVisible`(objectiveName, enable, playerID = PLAYER_1);

Описание

Задание *`objectiveName`* будет отображаться в интерфейсе игрока, если параметр *`enable`* равен `true`, и будет скрытым, если он равен `false`. Параметр *`playerID`* игнорируется для заданий, принадлежащих конкретному игроку. Для общих заданий, если параметр *`playerID`* задан, то он определяет игрока для которого нужно отобразить / скрыть задание, если нет — задание отображается / скрывается в интерфейсе первого игрока.

`objectiveName` — имя задания

`playerID` — идентификатор игрока (для заданий, принадлежащих конкретному игроку, параметр игнорируется, по умолчанию равен `PLAYER_1`).

SetObjectFlashlight

SetObjectFlashlight — цепляет к объекту карты точечный источник света

Синтаксис

`SetObjectFlashlight`(objectName, lightName);

Описание

Команда присоединяет к выбранному объекту карты дополнительный источник света. Список источников света которые можно использовать задается в описании карты (в поле *`resources->pointLights`*). Там же задаются их имена.

`objectName` — имя объекта.

`lightName` — имя источника света (задается в свойствах карты)

SetObjectOwner

SetObjectOwner — меняет принадлежность находящихся на карте объектов

Синтаксис

SetObjectOwner(objectName, playerID);

Описание

Меняет владельца объекта на карте. Менять принадлежность можно только для статических объектов способных иметь владельца и героев. Если герой находился в резерве, он будет из него удален. Героя нельзя сделать нейтральным.

objectName — имя объекта

playerID — идентификатор игрока — нового владельца объекта

SetObjectPosition

SetObjectPosition — перемещает объект

Синтаксис

SetObjectPosition(objectName, x, y, floor = -1);

Описание

Функция переставляет объект в выбранную позицию, задаваемую координатами тайла и номером уровня. Если при вызове функции не указывать значение параметра *floor*, по умолчанию будет использован уровень, на котором объект стоит в момент вызова. Перемещать можно только мобильные объекты. Если позиция, в которую нужно переместить объект недоступна, занята другим объектом или является интерактивным тайлом другого объекта, функция генерирует ошибку.

objectName — имя объекта

x, y — координаты тайла

floor — номер уровня (по умолчанию равен -1, что означает "тот же уровень")

SetPlayerResource

SetPlayerResource — установить количество ресурсов у игрока

Синтаксис

SetPlayerResource(player, resourceKind, quantity);

Описание

Устанавливает новое количество ресурсов вида *resourceKind* игроку *player*.

player — номер игрока от 1 до 8. Вместо чисел можно использовать глобальные константы PLAYER_1 до PLAYER_8.

resourceKind — число от 0 до 6, либо одна из констант WOOD, ORE, MERCURY, CRYSTAL, SULFUR, GEM, GOLD.

quantity не может быть отрицательным.

Команда работает для активных и проигравших игроков. Изменение количества ресурсов у игрока, не принимающего участия в игре, является ошибкой.

Ошибка

- если аргументы не прошли проверку на допустимость значений
- если указанный игрок в игре не участвует

SetPlayerStartResources

SetPlayerStartResources — установить стартовое количество ресурсов у игрока

Синтаксис

SetPlayerStartResources (player, wood, ore, mercury, crystal, sulfur, gem, gold) ;

Описание

Устанавливает новое количество стартовых ресурсов игроку *player*.

player — номер игрока от 1 до 8. Вместо чисел можно использовать глобальные константы PLAYER_1 до PLAYER_8.

wood, *ore*, *mercury*, *crystal*, *sulfur*, *gem*, *gold* — значения стартовых ресурсов, должны быть не меньше нуля.

Команда эквивалентна вызовам SetPlayerResource для каждого типа ресурса, за исключением влияния на результаты миссии (интерфейс результатов миссии).

Команда работает для активных и проигравших игроков. Изменение количества ресурсов у игрока, не принимающего участия в игре, является ошибкой.

Ошибка

- если аргументы не прошли проверку на допустимость значений
- если указанный игрок в игре не участвует

SetPlayerTeam

SetPlayerTeam — установить команду (team) игрока

Синтаксис

```
SetPlayerTeam( player, team );
```

Описание

Меняем команду игрока *player* на команду *team*. Все игроки одной команды (team) являются союзниками, другие игроки — их враги.

player — номер игрока от `1` до `8`. Вместо чисел можно использовать глобальные константы PLAYER_1 до PLAYER_8.

team — число от `1` до `8`.

Команда работает для любых игроков.

Ошибка

- если аргументы не прошли проверку на допустимость значений

SetRegionBlocked

SetRegionBlocked — устанавливает/снимает блокировку прохождения региона героями игроков

Синтаксис

```
SetRegionBlocked(regionName, status, playerID = -1);
```

Описание

Функция блокирует или разблокирует регион для прохождения его героями заданных игроков в зависимости от параметра *status*. Если параметр *playerID* задан — блокировка распространяется только на героев этого игрока, если нет — то на всех героев.

regionName — имя региона

status — флаг определяющий ставится блокировка или снимается

playerID — идентификатор игрока (по умолчанию равен -1, что означает — все игроки)

SetStandState

SetStandState — меняет состояние "стенда" на карте

Синтаксис

SetStandState(standName, stateID);

Описание

Функция меняет состояние "стенда" с заданным именем. При этом заменяется модель объекта и проигрываются заданный для данной смены состояний эффекты.

standName — имя стенда

stateID — номер состояния

SetTownBuildingLimitLevel

SetTownBuildingLimitLevel — задает ограничение на уровень постройки здания в городе

Синтаксис

SetTownBuildingLimitLevel(townName, buildingID, limit);

Описание

Функция задает ограничение на возможный уровень заданного здания в городе. Уровень 0 означает, что здание не может быть построено. Задаваемое ограничение должно быть не меньше, чем текущий уровень заданного здания.

townName — имя города

buildingID — идентификатор типа строения, список возможных значений находится в описании команды *GetTownBuildingLevel*.

limit — ограничение на постройку здания в городе

SetWarfogBehaviour

SetWarfogBehaviour — включить/выключить снятие тумана войны героями игрока.

Синтаксис

`SetWarfogBehaviour(`*onLand*, *onSea*`);`

Description

`onLand`(1/0) — будет ли герой игрока, перемещаясь *по суше*, снимать туман.

`onSea`(1/0) — будет ли герой игрока, перемещаясь *по воде*, снимать туман.

ShowFlyingSign

ShowFlyingSign — отображает отлетающее от объекта сообщение

Синтаксис

`ShowFlyingSign``(`messageName, objectName, targetPlayerID = -1, time = 1.0`);`

Описание

Функция отображает отлетающее от объекта с именем *`objectName`* сообщение. Сообщение задается по имени ресурса в базе (*`messageName`*). Сообщения можно выводить для одного игрока, или для всех сразу. Для показа сообщения конкретному игроку, параметр *`targetPlayerID`* должен содержать идентификатор этого игрока, для показа сообщения всем игрокам, он должен быть равным `-1`. Сообщение отлетает время *`time`* (по умолчанию — одну секунду).

`messageName` — имя сообщения в базе ресурсов

`objectName` — имя объекта, над которым появляется сообщение

`targetPlayerID` — идентификатор игрока, у которого появляется сообщение (`-1` для всех игроков)

`time` — время, на которое появляется сообщение

SiegeTown

SiegeTown — инициирует осаду города

Синтаксис

```
SiegeTown(heroName, townName, arenaName = "");
```

Описание

Инициирует осаду героем *heroName* города *townName*, причем в качестве имени города может быть задано как имя города на карте, так и указатель на ресурс с описанием города в базе ресурсов. И в том и в другом случае бой будет запущен так же, как и обычная осада города на карте, то есть будет использована такая информация о городе как его гарнизон, находящийся в городе герой, городской боевой скрипт и т.д. С помощью задания параметра *arenaName* можно заменить стандартное поле боя на любой другое.

heroName — имя героя, для которого нужно запустить сражение

townName — имя города, причем в качестве имени города может быть задано как имя города на карте, так и указатель на ресурс с описанием города в базе ресурсов.

arenaName — указатель на ресурс арены, на которой должен происходить бой (по умолчанию — "", что означает что бой будет проходить на стандартной для города арене)

StartCombat

StartCombat — запускает сражение с заданными параметрами

Синтаксис

```
StartCombat(,
            ,
            ,
            ,
            ,
            ,
            ,
            );
```

Описание

Запускает для героя *heroName* сражение с заданным набором существ и определенным героем (если он задан). Армия вражеского героя игнорируется. Есть возможность указать скрипт, который нужно запустить в начале сражения. Также есть возможность указать

функцию, которая будет вызвана в конце сражения и получит в качестве аргументов имя героя, для которого было запущено сражение и результат этого сражения.

heroName — имя героя, для которого нужно запустить сражение

enemyHeroName — имя вражеского героя (`nil` — если вражеского героя нет и бой идет только с существами)

creaturesCount — количество существ, с которыми будет производиться сражение

После параметра "количество существ" должно следовать *creaturesCount* пар вида

creatureType — тип существа

creatureAmount — количество существ этого типа

combatScriptName — имя скрипта, который будет запущен в начале сражения (по умолчанию — `nil`, что означает, что ни один скрипт запущен не будет)

combatFinishTrigger — имя функции (скрипта карты), которая будет запущена в конце сражения. Функция должна принимать два параметра (имя сражающегося героя и результат сражения (`nil` — если герой проиграл и `not nil` — если герой победил)).

arenaName — указатель на ресурс арены, на которой должен происходить бой (по умолчанию — nil, что означает что арена будет определяться по типу местности на которой стоит герой)

allowQuickCombat — если не `nil`, то сражение будет запущено в режиме быстрого боя, если этот режим включен в опциях игры и у героя *heroName* не отключена возможность быстрого боя (по умолчанию — nil, то есть запускается обычное сражение)

StartCutScene

StartCutScene — запускает ролик.

Синтаксис

`StartDialogScene``(cutSceneName, callback = "", saveName = "");`

Описание

Функция запускает заданный ролик.

cutSceneName — указатель на ролик в базе ресурсов

callback — имя функции, которая будет вызвана после проигрывания диалоговой сцены

saveName — имя save-файла (см. команду *Save*) который будет сделан перед диалоговой сценой.

StartDialogScene

StartDialogScene — запускает диалоговую сцену.

Синтаксис

StartDialogScene(dialogSceneName, callback = "", saveName = "");

Описание

Функция запускает заданную диалоговую сцену.

dialogSceneName — указатель на диалоговую сцену в базе ресурсов

callback — имя функции, которая будет вызвана после проигрывания диалоговой сцены

saveName — имя save-файла (см. команду *Save*) который будет сделан перед диалоговой сценой

Пример 1. Пример:

StartDialogScene("/DialogScenes/C6/M4/C1/DialogScene.xdb#xpointer(/DialogScene)")

StopPlaySound

StopPlaySound — останавливает проигрывание зацикленного звука

Синтаксис

StopPlaySound(loopingSoundID);

Описание

Останавливает проигрывание указанного циклического звука.

loopingSoundID — идентификатор звука, возвращается функциями *Play2DSound / Play3DSound* для циклических звуков.

StopVisualEffects

StopVisualEffects — останавливает визуальный эффекты на карте

Синтаксис

`StopVisualEffects`(tagName="");

Описание

Останавливает все визуальный эффекты с заданным tagName. Запускаются эффекты функцией *`PlayVisualEffect`*.

`tagName` — имя для управления эффектом, необязательный параметр — по умолчанию имя пустое (остановить все эффекты)

TeachHeroSpell

TeachHeroSpell — выдает герою заданное заклинание

Синтаксис

`TeachHeroSpell`(heroName, spell);

Описание

Функция делает попытку дать герою заклинание. Если герой уже знает его функция возвращает `nil`, в противном случае возвращается `not nil`.

`heroName` — имя героя

`spell` — идентификатор заклинания

TransformTown

TransformTown — меняет тип города на карте.

Синтаксис

`TransformTown`(townName, type);

Описание

Меняет тип города на карте на заданный. (В текущей реализации теряется вся информация о городе, кроме его имени и номера игрока, которому он принадлежит).

townName — имя города

type — новый тип города, может принимать одно из следующих значений:

TOWN_HEAVEN,

TOWN_PRESERVE,

TOWN_ACEDEMY,

TOWN_DUNGEON,

TOWN_NECROMANCY,

TOWN_INFERNO.

Trigger

Trigger — позволяет устанавливать и снимать функции-обработчики событий происходящих в мире

Синтаксис

Trigger(triggerType, ..., functionName);

Описание

Если параметр *functionName* не равен nil, функция устанавливает функцию-обработчик с этим именем на заданное событие игрового мира. В противном случае функция снимает обработчик с заданного события. Для каждого события может быть установлен только одна функция-обработчик. Установка второго обработчика на уже обрабатываемое событие снимает первый обработчик.

triggerType — тип события игрового мира, может принимать одно из следующих значений:

NEW_DAY_TRIGGER — начался новый день

WAR_FOG_ENTER_TRIGGER — герой вошел под туман (поведение тумана управляется командой *SetWarfogBehaviour*)

PLAYER_ADD_HERO_TRIGGER — у игрока появился новый герой

PLAYER_REMOVE_HERO_TRIGGER — игрок потерял героя

OBJECTIVE_STATE_CHANGE_TRIGGER — задание поменяло свое состояние

OBJECT_CAPTURE_TRIGGER — захвачен объект

OBJECT_TOUCH_TRIGGER — взаимодействие в интерактивным объектом

TOWN_HERO_DEPLOY_TRIGGER — выход героя из города

REGION_ENTER_AND_STOP_TRIGGER — герой зашел в регион (и должен в нем остановиться)

REGION_ENTER_WITHOUT_STOP_TRIGGER — герой зашел в регион (и не должен в нем останавливаться)

HERO_LEVELUP_TRIGGER — герой получил уровень

За типом события могут следовать один или несколько параметров, определяющие события какого именно мирового объекта нужно обрабатывать. Количество и смысл этих параметров зависят от типа события:

для NEW_DAY_TRIGGER — нет

для PLAYER_ADD_HERO_TRIGGER и PLAYER_REMOVE_HERO_TRIGGER — идентификатор игрока

для OBJECTIVE_STATE_CHANGE_TRIGGER — имя задачи

для OBJECT_CAPTURE_TRIGGER и OBJECT_TOUCH_TRIGGER — имя объекта на карте

для TOWN_HERO_DEPLOY_TRIGGER — имя города

для REGION_ENTER_AND_STOP_TRIGGER и REGION_ENTER_WITHOUT_STOP_TRIGGER — имя региона

для HERO_LEVELUP_TRIGGER — имя героя

functionName — имя функции — обработчика

В функцию обработчик при срабатывании триггера передаются следующие параметры:

NEW_DAY_TRIGGER – нет

WAR_FOG_ENTER_TRIGGER – имя героя

PLAYER_ADD_HERO_TRIGGER — имя героя

PLAYER_REMOVE_HERO_TRIGGER — имя героя и имя победившего его героя (nil — если герой был уволен)

OBJECTIVE_STATE_CHANGE_TRIGGER — идентификатор игрока, задание которого изменило статус

`OBJECT_CAPTURE_TRIGGER` – старый владелец объекта, новый владелец объекта, имя героя, его захватившего (если есть) и имя самого объекта

`OBJECT_TOUCH_TRIGGER` – имя героя и имя самого объекта

`TOWN_HERO_DEPLOY_TRIGGER` — имя героя

`REGION_ENTER_AND_STOP_TRIGGER` и `REGION_ENTER_WITHOUT_STOP_TRIGGER` – имя героя

`HERO_LEVELUP_TRIGGER` — нет

UnblockGame

UnblockGame — заблокирует интерфейс пользователя и работу AI.

Синтаксис

`UnblockGame``(void);`

Описание

Команда заблокирует интерфейс пользователя, работу AI и управление камерой, заблокированные командой *`BlockGame`*..

UnreserveHero

UnreserveHero — делает зарезервированного для определенного игрока героя снова доступным для остальных игроков.

Синтаксис

`UnreserveHero``(`heroName`);`

Описание

Герой перестает быть зарезервированным за определенным игроком и начинает учитываться при распределении героев по тавернам игроков в начале каждой недели.

Внимание: Если на момент вызова команды герой жив, он перестает быть зарезервированным, но не удаляется из списка героев, которыми владеет игрок. Доступным для других игроков он может стать таким же образом, как и обычный герой, принадлежащий какому-либо игроку (то есть: если умрет, сбежит с поля боя и не будет снова нанят до конца недели, будет уволен и т.д.). Если же на момент вызова команды

герой мертв, он сразу становится доступным для всех игроков, кроме того, который владел им до сих пор, так как считается проигравшим сражение.

heroName — имя зарезервированного за игроком героя.

Win

Win — при вызове функции игрок-человек выигрывает

Синтаксис

```
Win(void);
```

Описание

Данная функция предназначена для использования только в режиме одиночной игры. При вызове в сетевом режиме генерируется ошибка. После вызова функции игрок-человек немедленно выигрывает, все компьютерные игроки проигрывают.

Введение

Введение — общее описание скриптовой системы боев

Описание

Боевая скриптовая система состоит из двух основных частей — механизма обработки происходящих событий и механизма выполнения действий.

Для перехвата и обработки соответствующего события нужно определить функцию с некоторым предопределенным именем, соответствующим событию. Эта функция будет вызвана в тот момент, когда событие произойдет и ей будут переданы параметры, описывающие произошедшее событие. В случае если функция может замещать стандартный механизм работы в заданной ситуации, она должна возвращать значение, определяющее нужно ли производить по окончании работы функции стандартные действия соответствующие этой ситуации. Полный список событий которые можно перехватывать приводится в разделе Handlers.

Выполнение действий производится обычным образом с помощью вызова определенных команд. Полный список команд приводится в разделе Commands. Также существует старый набор команд, описанный в разделе Legacy, но его работоспособность и неизменность не гарантируется. Об использовании функций из этого набора просьба сообщать мне (dmitry.roubtsov@nival.com).

Prepare

Prepare — подготовка к сражению

Синтаксис

```
Prepare(void);
```

Описание

Функция вызывается перед началом расстановки игроками войск на поле боя. Расстановка войск начнется только после того как эта функция завершит свою работу.

Start

Start — начало сражения

Синтаксис

```
Start(void);
```

Описание

Функция вызывается после завершения расстановки игроками войск на поле боя и непосредственно перед началом сражения. Сражение начнется только после того как эта функция завершит свою работу.

AttackerHeroMove

AttackerHeroMove — ход героя атакующей стороны

Синтаксис

```
AttackerHeroMove(heroName);
```

Описание

Функция вызывается на ходу героя атакующей стороны и принимает его имя (боевого объекта). Это имя затем можно использовать для выполнения определенных действий с участием этого героя (например, сотворить от его имени заклинание). Функция должно возвращать значение nil если после действий выполненных функцией должны быть

выполнены стандартные для этого героя действия (под управлением AI) и not nil — если нет.

DefenderHeroMove

DefenderHeroMove — ход героя защищающейся стороны

Синтаксис

DefenderHeroMove(heroName);

Описание

Функция вызывается на ходу героя защищающейся стороны и принимает его имя (боевого объекта). Это имя затем можно использовать для выполнения определенных действий с участием этого героя (например, сотворить от его имени заклинание). Функция должно возвращать значение nil если после действий выполненных функцией должны быть выполнены стандартные для этого героя действия (под управлением AI) и not nil — если нет.

AttackerCreatureMove

AttackerCreatureMove — ход существа атакующей стороны

Синтаксис

AttackerCreatureMove(creatureName);

Описание

Функция вызывается на ходу существа атакующей стороны и принимает его имя. Это имя затем можно использовать для выполнения определенных действий с участием этого существа. Функция должно возвращать значение nil если после действий выполненных функцией должны быть выполнены стандартные для этого существа действия (под управлением ai) и not nil если нет.

DefenderCreatureMove

DefenderCreatureMove — ход существа защищающейся стороны

Синтаксис

DefenderCreatureMove(creatureName);

Описание

Функция вызывается на ходу существа защищающейся стороны и принимает его имя. Это имя затем можно использовать для выполнения определенных действий с участием этого существа. Функция должно возвращать значение nil если после действий выполненных функцией должны быть выполнены стандартные для этого существа действия (под управлением ai) и not nil если нет.

AttackerWarMachineMove

AttackerWarMachineMove — ход осадной машины атакующей стороны

Синтаксис

`AttackerWarMachineMove`(warMachineName);

Описание

Функция вызывается на ходу осадной машины атакующей стороны и принимает ее имя. Функция должна возвращать значение nil если после действий выполненных функцией должны быть выполнены стандартные для этой машины действия (под управлением ai) и not nil если нет.

DefenderWarMachineMove

DefenderWarMachineMove — ход осадной машины защищающейся стороны

Синтаксис

`DefenderWarMachineMove`(warMachineName);

Описание

Функция вызывается на ходу осадной машины защищающейся стороны и принимает ее имя. Функция должна возвращать значение nil если после действий выполненных функцией должны быть выполнены стандартные для этой машины действия (под управлением ai) и not nil если нет.

DefenderBuildingMove

DefenderBuildingMove — ход здания защищающейся стороны

Синтаксис

`DefenderBuildingMove`(buildingName);

Описание

Функция вызывается на ходу здания защищающейся стороны и принимает его имя. Функция должна возвращать значение nil если после действий выполненных функцией должны быть выполнены стандартные для этой машины действия (под управлением ai) и not nil если нет.

AttackerCreatureDeath

AttackerCreatureDeath — смерть существа атакующей стороны

Синтаксис

`AttackerCreatureDeath`(creatureName);

Описание

Функция вызывается в случае смерти существа атакующей стороны и принимает его имя.

DefenderCreatureDeath

DefenderCreatureDeath — смерть существа защищающейся стороны

Синтаксис

`DefenderCreatureDeath`(creatureName);

Описание

Функция вызывается в случае смерти существа защищающейся стороны и принимает его имя.

AttackerWarMachineDeath

AttackerWarMachineDeath — разрушение осадной машины атакующей стороны

Синтаксис

`AttackerWarMachineDeath`(warMachineName);

Описание

Функция вызывается в случае разрушения осадной машины атакующей стороны и принимает его имя.

DefenderWarMachineDeath

DefenderWarMachineDeath — разрушение осадной машины защищающейся стороны

Синтаксис

`DefenderWarMachineDeath`(warMachineName);

Описание

Функция вызывается в случае разрушения осадной машины защищающейся стороны и принимает его имя.

DefenderBuildingDeath

DefenderBuildingDeath — разрушение строения защищающейся стороны

Синтаксис

`DefenderBuildingDeath`(buildingName);

Описание

Функция вызывается в случае разрушения строения защищающейся стороны и принимает его имя.

IsHuman

IsHuman — определяет играет ли за определенную сторону человек

Синтаксис

`IsHuman`(side);

Описание

Функция возвращает `not nil` если заданная сторона участвующая в бою принадлежит игроку-человеку (независимо от того управляет ли он ее действиями сам или управление ее действиями происходит автоматически) и `nil` если нет.

`side` — номер стороны, может быть `ATTACKER` или `DEFENDER`.

IsComputer

IsComputer — определяет играет ли за определенную сторону компьютер

Синтаксис

`IsComputer`(side);

Описание

Функция возвращает `not nil` если заданная сторона участвующая в бою принадлежит игроку-компьютеру и `nil` если нет.

`side` — номер стороны, может быть `ATTACKER` или `DEFENDER`.

SetControlMode

SetControlMode — задает режим управления боем для стороны игрока-человека

Синтаксис

`SetControlMode`(side, mode);

Описание

Команда управляет включением / выключением режима автоматического боя и установкой / снятием блокировки ручного изменения игроком этого режима. Сторона должна управляться игроком-человеком.

`side` — номер стороны, может быть `ATTACKER` или `DEFENDER`.

`mode` — номер режима, в зависимости от него выполняется одно из следующих действий:

`MODE_NORMAL` — состояние режима автоматического боя не меняется, блокировка на его переключение снимается

`MODE_MANUAL` — режим автоматического боя выключается, устанавливается блокировка на его переключение игроком

`MODE_AUTO` — включается режим автоматического боя, устанавливается блокировка на его переключение игроком

EnableDynamicBattleMode

EnableDynamicBattleMode — включение/отключение режима «Dynamic Battle»

Синтаксис

`EnableDynamicBattleMode`(enable);

Описание

Включение/отключение режима «Dynamic Battle».

EnableAutoFinish

EnableAutoFinish — включает/выключает стандартный механизм проверки окончания боя

Синтаксис

`EnableAutoFinish`(enable);

Описание

Параметр передаваемый в команду определяет будет ли закончен бой, если у одной из сторон не останется войск. Если параметр равен `nil` бой в этом случае не закончится. Изначально этот режим включен.

Finish

Finish — заканчивает бой

Синтаксис

`Finish`(winnerSide);

Описание

Команда заканчивает бой, причем сторона *`winnerSide`* считается победившей.

`winnerSide` — сторона победившая в бою, может быть `ATTACKER` или `DEFENDER`.

Break

Break — прерывает бой

Синтаксис

`Break(void);`

Описание

Команда прерывает бой. При этом не показывается экран результатов боя и никакого эффекта на игру бой не оказывает (не дается опыт, не захватываются здания и т.д.).

GetAttackerHero

GetAttackerHero — возвращает героя атакующей стороны

Синтаксис

```
GetAttackerHero(void);
```

Описание

Функция возвращает имя героя атакующей стороны или `nil` если его нет.

GetAttackerCreatures

GetAttackerCreatures — возвращает существ атакующей стороны

Синтаксис

```
GetAttackerCreatures(void);
```

Описание

Функция возвращает массив имен существ атакующей стороны.

GetAttackerWarMachines

GetAttackerWarMachines — возвращает осадные машины атакующей стороны

Синтаксис

```
GetAttackerWarMachines(void);
```

Описание

Функция возвращает массив имен осадных машин атакующей стороны.

GetAttackerWarMachine

GetAttackerWarMachine — возвращает осадную машину атакующей стороны заданного типа

Синтаксис

```
GetAttackerWarMachine(type);
```

Описание

Функция возвращает имя осадной машины атакующей стороны заданного типа или если `nil` ее нет.

`type` — тип осадной машины, может принимать одно из следующих значений:

`WAR_MACHINE_BALLISTA` — баллиста

`WAR_MACHINE_CATAPULT` — катапульта

`WAR_MACHINE_FIRST_AID_TENT` — палатка

`WAR_MACHINE_AMMO_CART` — тележка с боеприпасами

GetDefenderHero

GetDefenderHero — возвращает героя защищающейся стороны

Синтаксис

```
GetDefenderHero(void);
```

Описание

Функция возвращает имя героя защищающейся стороны или `nil` если его нет.

GetDefenderCreatures

GetDefenderCreatures — возвращает существ защищающейся стороны

Синтаксис

```
GetDefenderCreatures(void);
```

Описание

Функция возвращает массив имен существ защищающейся стороны.

GetDefenderWarMachines

GetDefenderWarMachines — возвращает осадные машины защищающейся стороны

Синтаксис

```
GetDefenderWarMachines(void);
```

Описание

Функция возвращает массив имен осадных машин защищающейся стороны.

GetDefenderWarMachine

GetDefenderWarMachine — возвращает осадную машину защищающейся стороны заданного типа

Синтаксис

```
GetDefenderWarMachine(type);
```

Описание

Функция возвращает имя осадной машины защищающейся стороны заданного типа или если `nil` ее нет.

`type` — тип осадной машины, может принимать одно из следующих значений:

`WAR_MACHINE_BALLISTA` — баллиста

`WAR_MACHINE_CATAPULT` — катапульта

`WAR_MACHINE_FIRST_AID_TENT` — палатка

`WAR_MACHINE_AMMO_CART` — тележка с боеприпасами

GetDefenderBuildings

GetDefenderBuildings — возвращает строения защищающейся стороны

Синтаксис

```
GetDefenderBuildings(void);
```

Описание

Функция возвращает массив имен строений защищающейся стороны.

GetDefenderBuilding

GetDefenderBuilding — возвращает строение защищающейся стороны заданного типа

Синтаксис

```
GetDefenderBuilding(type);
```

Описание

Функция возвращает имя строения защищающейся стороны заданного типа или если `nil` его нет.

`type` — тип строения, может принимать одно из следующих значений:

`BUILDING_WALL` — стена

`BUILDING_GATE` — ворота

`BUILDING_LEFT_TOWER` — левая башня

`BUILDING_BIG_TOWER` — центральная башня

`BUILDING_RIGHT_TOWER` — правая башня

`BUILDING_MOAT` — ров

IsAttacker

IsAttacker — определяет принадлежит ли объект к атакующей стороне

Синтаксис

`IsAttacker``(`unitName`);`

Описание

Функция возвращает `not nil` если объект принадлежит к атакующей стороне и `nil` если нет.

`unitName` — имя объекта

IsDefender

IsDefender — определяет принадлежит ли объект к защищающейся стороне

Синтаксис

`IsDefender``(`unitName`);`

Описание

Функция возвращает `not nil` если объект принадлежит к защищающейся стороне и `nil` если нет.

`unitName` — имя объекта

IsHero

IsHero — определяет является ли объект героем

Синтаксис

`IsHero(`unitName`);`

Описание

Функция возвращает `not nil` если объект является героем и `nil` если нет.

`unitName` — имя объекта

IsCreature

IsCreature — определяет является ли объект существом

Синтаксис

`IsCreature(`unitName`);`

Описание

Функция возвращает `not nil` если объект является существом и `nil` если нет.

`unitName` — имя объекта

IsWarMachine

IsWarMachine — определяет является ли объект осадной машиной

Синтаксис

`IsWarMachine(`unitName`);`

Описание

Функция возвращает `not nil` если объект является осадной машиной и `nil` если нет.

`unitName` — имя объекта

IsBuilding

IsBuilding — определяет является ли объект строением

Синтаксис

`IsBuilding(`unitName`);`

Описание

Функция возвращает `not nil` если объект является строением и `nil` если нет.

`unitName` — имя объекта

GetHeroName

GetHeroName — возвращает имя героя

Синтаксис

`GetHeroName(`unitName`);`

Описание

Функция возвращает настоящее (не боевое) имя героя.

`unitName` — имя объекта

GetCreatureType

GetCreatureType — возвращает тип существа

Синтаксис

`GetCreatureType(`unitName`);`

Описание

Функция возвращает тип существа.

`unitName` — имя объекта

GetCreatureNumber

GetCreatureNumber — возвращает количество существ в группе

Синтаксис

`GetCreatureNumber(`unitName`);`

Описание

Функция возвращает количество существ в группе.

unitName — имя объекта

GetWarMachineType

GetWarMachineType — возвращает тип осадной машины

Синтаксис

```
GetWarMachineType(unitName);
```

Описание

Функция возвращает возвращает тип осадной машины. Может возвращать одно из следующих значений:

WAR_MACHINE_BALLISTA — баллиста

WAR_MACHINE_CATAPULT — катапульта

WAR_MACHINE_FIRST_AID_TENT — палатка

WAR_MACHINE_AMMO_CART — тележка с боеприпасами

unitName — имя объекта

GetBuildingType

GetBuildingType — возвращает тип строения

Синтаксис

```
GetBuildingType(unitName);
```

Описание

Функция возвращает возвращает тип строения. Может возвращать одно из следующих значений:

BUILDING_WALL — стена

BUILDING_GATE — ворота

`BUILDING_LEFT_TOWER` — левая башня

`BUILDING_BIG_TOWER` — центральная башня

`BUILDING_RIGHT_TOWER` — правая башня

`BUILDING_MOAT` — ров

`unitName` — имя объекта

GetUnitPosition

GetUnitPosition — возвращает координаты объекта

Синтаксис

`GetUnitPosition(`unitName`);`

Описание

Функция возвращает координаты *`x`* и *`y`* местоположения объекта.

`unitName` — имя объекта

AddCreature

AddCreature — добавляет одной из сторон существ заданного типа

Синтаксис

`AddCreature(`side, type, number, x `=` -1, y `=` -1`);`

Описание

Функция добавляет стороне *`side`* существ типа *`type`* в количестве *`number`* штук. Если параметры *`x`* и *`y`* заданы существо возникает в ближайшей свободной к `(x,y)` клетке, в противном случае существо появляется в случайной точке арены. Также можно задавать только одну из координат клетки. Присоединенные этой командой существа остаются в армии героя после боя (в том случае, если они туда помещаются).

`side` — сторона на которую призывается существо, может принимать одно из двух значений:

`ATTACKER` — атакующая сторона

`DEFENDER` — защищающаяся сторона

`type` — тип существа

`number` — количество

`x, y` — координаты клетки в которой должно появиться существо (значение `-1` означает произвольное значение координаты)

SummonCreature

SummonCreature — призывает существ заданного типа

Синтаксис

`SummonCreature(`side, type, number, x `=` -1, y `=` -1`);`

Описание

Функция призывает на сторону *`side`* существ типа *`type`* в количестве *`number`* штук. Если параметры *`x`* и *`y`* заданы существо возникает в ближайшей свободной к `(x,y)` клетке, в противном случае существо появляется в случайной точке арены. Также можно задавать только одну из координат клетки.

`side` — сторона на которую призывается существо, может принимать одно из двух значений:

`ATTACKER` — атакующая сторона

`DEFENDER` — защищающаяся сторона

`type` — тип существа

`number` — количество

`x, y` — координаты клетки в которой должно появиться существо (значение `-1` означает произвольное значение координаты)

GetUnitManaPoints

GetUnitManaPoints — возвращает количество очков маны объекта

Синтаксис

`GetUnitManaPoints(`unitName`);`

Описание

Функция возвращает текущее количество очков маны объекта.

`unitName` — имя объекта

GetUnitMaxManaPoints

GetUnitMaxManaPoints — возвращает максимальное возможное количество очков маны объекта

Синтаксис

`GetUnitMaxManaPoints(`unitName`);`

Описание

Функция возвращает максимальное возможное количество очков маны объекта.

`unitName` — имя объекта

SetUnitManaPoints

SetUnitManaPoints — устанавливает количество очков маны объекта

Синтаксис

`SetUnitManaPoints(`unitName, manaPoints`);`

Описание

Функция меняет количество оставшихся очков маны объекта. Можно задавать значения выше чем максимально допустимые обычно для объекта.

`unitName` — имя объекта

`manaPoints` — количество очков маны

UnitCastAimedSpell

UnitCastAimedSpell — колдует заклинание действующее на определенный объект на поля боя

Синтаксис

`UnitCastAimedSpell(`unitName, spell, targetUnit`);`

Описание

Функция колдует заклинание действующее на определенный объект на поля боя от имени заданного объекта.

`unitName` — имя объекта

`spell` — код заклинания

`targetName` — имя объекта, который будет подвергнут действию заклинания

UnitCastAreaSpell

UnitCastAreaSpell — колдует заклинание действующее на определенную область поля боя

Синтаксис

`UnitCastAreaSpell(`unitName, spell, x, y`);`

Описание

Функция колдует заклинание заклинание действующее на определенную область поля боя от имени заданного объекта.

`unitName` — имя объекта

`spell` — код заклинания

`x, y` — координаты центра области, на ноторую подействует заклинание

UnitCastGlobalSpell

UnitCastGlobalSpell — колдует ненаправленное заклинание

Синтаксис

`UnitCastGlobalSpell(`unitName, spell`);`

Описание

Функция колдует заклинание действующее на всех находящихся на поле боя от имени заданного объекта.

`unitName` — имя объекта

`spell` — код заклинания

EnableCinematicCamera

EnableCinematicCamera — отключение/включение кинематографической камеры в комбате

Синтаксис

`EnableCinematicCamera``(`enable`);`

Описание

Работает только для текущего комбата. Теперь камера может переключаться на кинематографическую только если она разрешена в настройках и не запрещена скриптом.

---

addUnit

addUnit — добавить существо (отряд существ)

Синтаксис

`addUnit``(`unitID, side, tileX, tileY, quantity, name`);`

Описание

Создаёт отряд существ на боевой арене в тайле с указанными координатами.

unitID — существо для постановки на поле боя, *side* — атакующая сторона или защищающаяся (0 или 1), *tileX* и *tileY* — координаты существа, *quantity* — количество существ в отряде, *name* — имя для обращения из скрипта.

combatEnableFinish

combatEnableFinish — включить/выключить штатное завершение боя

Синтаксис

combatEnableFinish(enable);

Описание

В штатном режиме (конечная игра) бой автоматически заканчивается, когда одна из армий полностью уничтожена; экран боя закрывается, игра показывает экран карты. При разработке и для демонстрационных версий игры бывает необходимо оставаться на экране боя даже тогда, когда одна из армий разбита (например, для того, чтобы её восстановить -- добавить проигравшему герою ещё войск).

Команда включает/выключает штатную реакцию (завершение боя) на полное уничтожение одной из армий.

---

combatPlayEmotion

combatPlayEmotion — отыграть "эмоцию" победы или поражения какой-либо армией

Синтаксис

combatPlayEmotion(side, victory);

Описание

Команда отыгрывает "эмоцию" победы или поражения (анимации "happy" или "sorrow" на всех существах) для указанной армии.

side — ATTACKER или DEFENDER

victory — true, если надо отыграть победу, false — поражение

---

combatReadyPerson

combatReadyPerson — выдает существо готовое к действию или nil если такого нет

Синтаксис

```
combatReadyPerson(void);
```

Описание

[добавить]

---

combatStarted

combatStarted — готов ли уже комбат к выполнению команд

Синтаксис

```
combatStarted(void);
```

Описание

Возвращает не nil если комбат уже стартовал, nil — если еще не стартовал.

[добавить]

---

commandDoSpecial

commandDoSpecial — дать существу команду применить спец. свойство/возможность

Синтаксис

```
commandDoSpecial(name, abilityID, targetX = -1, targetY = -1, use-game-logic-
limits = nil);
```

Описание

[добавить]

commandDoSpell

commandDoSpell — дать существу команду сотворить заклинание

Синтаксис

commandDoSpell(name, spellID, targetX = -1, targetY = -1, use-game-logic-limits = nil);

Описание

[добавить]

---

commandDefend

commandDefend — дать существу команду стать в оборону

Синтаксис

commandDefend(name, use-game-logic-limits = nil);

Описание

[добавить]

---

commandMove

commandMove — дать существу команду идти в другое место

Синтаксис

commandMove(name, tileX, tileY, use-game-logic-limits = nil);

Описание

[добавить]

---

commandMoveAttack

commandMoveAttack — дать существу команду атаковать ближней атакой

Синтаксис

commandMoveAttack(name, targetName, use-game-logic-limits = nil);

Описание

[добавить]

---

commandShot

commandShot — дать существу команду атаковать дальней атакой

Синтаксис

commandShot(name, targetName, use-game-logic-limits = nil);

Описание

[добавить]

---

displace

displace — переставить юнит на другое место арены

Синтаксис

displace(name, tileX, tileY);

Описание

Переносит отряд существ с именем *name* на клетку арены (*tileX*, *tileY*).

tileX и *tileY* должны попадать в арену, клетка (*tileX*, *tileY*) не должена быть блокированной. Само переставляемое существо блокирует клетки, на которых находится в момент перед перемещением, таким образом большие существа (размером 2x2 клетки) не могут быть переставлены ближе, чем на 2-е клетки от текущего положения.

EnableRealtime

EnableRealtime — устанавливает режим боя (turnbased или realtime).

Синтаксис

`EnableRealtime``(enable);`

Описание

Команда устанавливает режим боя "realtime", если параметр равен `true`, и "turnbased", если парамент равен `false`.

---

exist

exist — есть ли существо на арене?

Синтаксис

`bool` **`exist`**`(name);`

Описание

[добавить]

IsCombatRealtime

IsCombatRealtime — определяет режим боя (turnbased или realtime).

Синтаксис

`IsCombatRealtime``(void);`

Описание

Функция возвращает `true`, если бой происходит в реальном времени и `false`, если бой пошаговый .

---

playAnimation

playAnimation — проиграть анимацию на существе

Синтаксис

`playAnimation`(unitName, animName, action);

Описание

Проигрывает на указанном существе анимацию *`animName`*. Анимация проигрывается таким же точно образом (*`action`*), каким такое действие проигрывается в игре.

`action` определяет, как именно анимация будет играться. В игре выделены следующие виды действий:

- `INVISIBLE` — (особое) объект становится невидимым
- `IDLE` — анимация играется циклически до тех пор, пока не будет заменена какой-либо другой анимацией
- `ONESHOT_STILL` — анимация проигрывается однократно, по окончании объект остаётся в позе последнего кадра анимации
- `ONESHOT` — анимация проигрывается однократно, по окончании для объекта запускается его стандартная idle-анимация
- `MOVE` — анимация проигрывается циклически по пути в пространстве, по окончании – стандартная idle-анимация
- `NON_ESSENTIAL` — то же что `ONESHOT`, но анимация играется только в том случае, если объект не "занят", то есть если на объекте не играется ничего, кроме `IDLE`-анимации

`animName` — символьное имя анимации ("idle00", "attack00", "hit" и т.п.), его следует знать заранее. Если у юнита анимации с именем *`animName`* нет, то команда ничего не делает.

Действие `MOVE` данной командой не поддерживается.

---

pos

pos — координаты существа на арене по его имени

Синтаксис

`x,y =` **`pos`**(name);

Описание

Возвращает два числа, соответственно, `x` и `y` координаты указанного отряда существ на арене.

Может быть использована непосредственно в вызове команд, принимающих численные координаты, например:

```
commandDoSpecial("archers", pos("imps-2"), scatter-shot);
```

вместо:

```
commandShot("archers", 5, 7); // 5,7 -- координаты imps-2
```

или

```
x,y = pos("imps-2");
commandShot("archers", x, y);
```

[добавить]

RemoveAllUnits

RemoveAllUnits — убрать все существа с арены

Синтаксис

RemoveAllUnits(void);

Описание

Убирает всех существ с арены

---

removeUnit

removeUnit — убрать отряд существ

Синтаксис

removeUnit(name);

Описание

[добавить]

---

setATB

setATB — установить ATB значение для существа

Синтаксис

setATB(name, newATB);

Описание

[добавить]

---

combatSetPause

combatSetPause — установить или снять паузу в комбате

Синтаксис

combatSetPause(pause);

Описание

[добавить]

---

showGrid

showGrid — включить/выключить отображение сетки тайлов

Синтаксис

showGrid(enable = 1);

Описание

Включает отображение тайловой сетки на арене, когда *enable* – истина (всё, кроме `nil`).

Выключает отображение сетки, когда *enable* – ложь (`nil`).

---

showHighlighting

showHighlighting — включить/выключить подсветку возможного хода

Синтаксис

```
showHighlighting(enable = 1);
```

Описание

Включает отображение подсветки тайлов арены, когда *enable* – истина (всё, кроме `nil`).

Выключает отображение подсветки, когда *enable* – ложь (`nil`).

Когда подсветка включена, для текущего выбранного существа спец.образом выделяются все тайлы арены, до которых существо может дойти.

---

unitNames

unitNames — выдать список имён существ на арене

Синтаксис

```
list = unitNames(void);
```

Описание

Возвращает строку со списком имён существ, находящихся в данный момент на арене. Имена существ разделяются пробелом.

Возвращаются имена всех существ, независимо от их состояния.

`print(unitNames())` выводит список имён существ в консоль.

---

wait

wait — дождаться выполнения всех текущих действий

Синтаксис

```
wait(void);
```

Описание

[добавить]

IsTutorialItemEnabled

IsTutorialItemEnabled — узнать, будет ли выведен элемент tutorial-а

Синтаксис

`IsTutorialItemEnabled`(name);

Описание

name — имя элемента tutorial-а

IsTutorialMessageBoxOpen

IsTutorialMessageBoxOpen — узнать, открыто ли в данный момент какое-то окошко с tutorial текстом

Синтаксис

`IsTutorialMessageBoxOpen`();

Описание

Возвращает `true`, если на экране присутствует выведенное функцией `TutorialMessageBox` окно.

TutorialActivateHint

TutorialActivateHint — включает триггер на появление конкретного окна, при срабатывании выводится окно с tutorial текстом

Синтаксис

`TutorialActivateHint`(stringID);

Описание

Находит в UIConsts.tutorialOptions.hints элемент-дескриптор по строковому идентификатору *`stringID`*. Когда в следующий раз откроется окно, указанное в дескрипторе, будет показано окошко с текстом (такое же, как выводимое функцией `TutorialMessageBox`).

TutorialMessageBox

TutorialMessageBox — показать окошко с tutorial текстом

Синтаксис

`TutorialMessageBox`(*stringID*);

Описание

Находит в UIConsts.tutorialOptions.messages элемент по строковому идентификатору *`stringID`* и выводит окно с текстом, на который ссылается этот элемент.

TutorialSetBlink

TutorialSetBlink — включить/выключить подсветку контрола

Синтаксис

`TutorialSetBlink`(*stringID*, *turnOn*);

Описание

Находит в UIConsts.tutorialOptions.blinks элемент-дескриптор по строковому идентификатору *`stringID`*, и включает/выключает подсветку (в зависимости от значения *`turnOn`*)

---

toggleTutorialMode

toggleTutorialMode — показать/спрятать пользовательский интерфейс экрана tutorial

Синтаксис

`toggleTutorialMode`(enable = 1);

Описание

Включает отображение элементов пользовательского интерфейса, когда *`enable`* – истина (всё, кроме `nil`).

Выключает отображение элементов интерфейса, когда *`enable`* – ложь (`nil`).

---

showMessage

showMessage — показать текст tutorial-сообщения

Синтаксис

`showMessage``(`textID`);`

Описание

Выводит текст в предназначенном для этого элементе пользовательского интерфейса.

`textID` – численный идентификатор текста в игровой базе.

Если интерфейс экрана выключен, то текст сообщения не будет видимым до тех пор, пока интерфейс экрана не будет включён.

---

clearMessage

clearMessage — убрать текст tutorial-сообщения

Синтаксис

`clearMessage``(void);`

Описание

Очищает от текста предназначенный для вывода tutorial-сообщений элемент пользовательского интерфейса.

---

changeSubject

changeSubject — сменить субъект презентации

Синтаксис

```
changeSubject(unitID);
```

Описание

Устанавливает или меняет существо, для которого должна вестись презентация.

unitID — численный идентификатор существа в игровой базе.

---

subject

subject — вывести в консоль текущий субъект презентации

Синтаксис

```
subject(void);
```

Описание

Выводит в консоль DBID существа, являющегося предметом текущей презентации.

---

shots

shots — вывести в консоль список кадров текущей презентации

Синтаксис

```
shots(void);
```

Описание

Выводит в консоль имена и порядковые номера кадров, имеющихся в текущей презентации.

---

shotsNumber

shotsNumber — определить количество кадров текущей презентации

Синтаксис

```
int shotsNumber(void);
```

Описание

Возвращает количество кадров текущей презентации.

Введение

Введение — общее описание городской скриптовой системы

Описание

На данный момент система способна только обрабатывать происходящие события. Для перехвата и обработки соответствующего события нужно определить функцию с некоторым предопределенным именем, сообтветствующим событию. Эта функция будет вызвана в тот момент, когда событие произойдет и ей будут переданы параметры описывающие произошедшее событие. Полный список событий которые можно перехватывать приводится в разделе Handlers.

HeroHired

HeroHired — найм героя

Синтаксис

```
HeroHired(name);
```

Описание

Функция вызывается в случае найма героя игроком.

CreatureHired

CreatureHired — найм существа

Синтаксис

`CreatureHired(`type, number`);`

Описание

Функция вызывается в случае найма игроком войск.

GetObjectNamesByType

GetObjectNamesByType — Получает все объекты заданного типа

Синтаксис

`GetObjectNamesByType``(`objectTypeName`);`

Описание

Получает все объекты заданного типа. Для зданий нужно задавать тип(*`objectTypeName`*), прописанный в *`Shared-Type`* Для монстров — это `Creature` Для героев — *`HeroClass`* Для городов — *`TownRace`* Можно задавать подстроки для задания более общих типов. Например: `BUILDING` — все здания `ABANDONED_MINE` или `BUILDING_ABANDONED_MINE` — для заброшенных шахт `DWELLING` — для всех двеллингов `DWELLING_<тип двеллинга>` — конкретные двеллинги `CREATURE` — все монстры `CREATURE_PEASANT` или просто `PEASANT` — крестьяне `HERO` — все герои `HERO_CLASS_WIZARD` — все герое Академии `TOWN` — все города `TOWN_HEAVEN` — все города расы `HEAVEN`

GetGuardsTier

GetGuardsTier — Получает массив целых чисел — tier'ов охраны объекта (объекты типов flagged & guardable).

Синтаксис

`GetGuardsTier``(`objectName`);`

Описание

Получает массив целых чисел — tier'ов охраны объекта (объекты типов flagged & guardable). (На самом деле возвращается не совсем tier, а tier*2 + upgraded).

`objectName` — скриптовое имя объекта.

GetMonsterTier

GetMonsterTier — Получает целое число — tier монстра.

Синтаксис

```
GetMonsterTier(monsterName);
```

Описание

Получает целое число — tier монстра. На самом деле возвращается не совсем tier, а tier*2 + upgraded).

monsterName — скриптовое имя объекта.

GetHeroMovePoints

GetHeroMovePoints — Получает количество MovePoints (в которых измеряется стоимость пути по карте)

Синтаксис

```
GetHeroMovePoints (heroName);
```

Описание

Получает количество MovePoints (в которых измеряется стоимость пути по карте)).

heroName — скриптовое имя героя.

GetTerrainSize

GetTerrainSize — Функция возвращает 2 числа — размер террейна карты по x и y

Синтаксис

```
GetTerrainSize(void);
```

Описание

Функция возвращает 2 числа — размер террейна карты по *x* и *y*

IsTilePassable

IsTilePassable — Функция проверяет, проходим ли для героев тайл с координатами x, y на этаже nFloor.

Синтаксис

`IsTilePassable`(x, y, nFloor);

Описание

Функция проверяет, проходим ли для героев тайл с координатами *x, y* на этаже *nFloor*.

IsTileReachable

IsTileReachable — Функция проверяет, достижим ли для героя тайл с координатами x, y на этаже nFloor.

Синтаксис

`IsTileReachable`(heroName, x, y, nFloor);

Описание

Функция проверяет, достижим ли для героя *heroName* тайл с координатами *x, y* на этаже *nFloor*.