Скрипты TES 4 Oblivion для модмейкеров

Oblivion Scripting for Modmaker

OSFM v1.0

ТОМ 1

Базовый курс скриптинга TES 4 Oblivion

OSFM Team
2006

Оглавление

1. Предисловие к первому изданию

Основной целью учебника является написание максимально полного и качественного руководства по скриптовому языку программирования для игры TES 4 Oblivion. Учебник предназначен для модмейкеров - людей, создающих свои собственные модификации игры - как для начинающих, так и для опытных знатоков-скриптологов.

Немного истории:
Вы, конечно же, знаете о великом учебнике для игры TES 3 Morrowind - MSfD (Morrowind Scripting for Dummies). Автор учебника - GhanBuriGhan. Последняя авторская версия - MSfD 8.0. По MSfD 8.0 училось писать скрипты множество людей и для многих этот учебник действительно стал настольной книгой.

Начиная с 5-й версии началась работа над переводом MSfD на русский язык. В работе участвовали Boblen (в настоящий момент член команды OSFM team), Falca, Amargo (Nomad), Kuja, Ladimir, Vlad kudriashov, Turin Turambar и многие другие. Пока длились работы над переводом, версия MSfD выросла до седьмой. В конце концов была выпущена русская оф-лайн версия MSfD v7.0 rus. Немного раньше до этого знаменательного события в интернете появилась также русская он-лайн версия MSfD v7.0, автором перевода которой стал Aidan4.
Последнюю, 8-ю версию, перевел Gwathlobal, который, к слову, также является членом нашей команды. В дальнейшем Gwathlobal опубликовал еще одну русскую версию - MSfD 8.1 rus, в которую внес собственные дополнения. Учебник запакован в архив 7z и имеет небольшой размер - 440кВ.
Скачать его вы можете на сайте TESPlay:
http://tesplay.ru/content/articles/articles_m.shtml?view=5

Появилась информация, что сейчас идет работа над новой, 9-й версией MSfD. На сей раз эстафету от GhanBuriGhan принял новый автор - Yacoby. Отслеживать, как у него обстоят дела, вы можете на официальном форуме в теме "Morrowind Scripting for Dummies 9": http://www.elderscrolls.com/forums/index.p...howtopic=553402
Ну, что же, пожелаем ему усидчивости и терпения, поскольку путь к релизу ох как нелегок...

Предпосылки для написания нового учебника - OSFM.
В последнюю игру - TES 4 Oblivion, по сравнению с TES 3 Morrowind, было внесено множество изменений, касающихся практически всех аспектов геймплея. И это, конечно, повлекло за собой неизбежные изменения в принципах создания модификаций к основной игре. Коснулось это и такого очень важного вопроса, как написание скриптов. Скрипты используются непосредственно в игре и пишутся на специальном интерпретируемом языке программирования. Но вот как это делается, каковы особенности языка, функций, команд - увы, на момент релиза игры информации на русском языке было очень мало, не лучше обстоят дела и сейчас, спустя почти год...
Эта ситуация и стала главной предпосылкой к написанию данного учебника.

Реализация проекта OSFM.
Проект получил название OSFM и стартовал в конце мая 2006 года. OSFM - аббревиатура, которая расшифровывается так - "Oblivion Scripting for Modmakers", что в русском переводе можно перевести как "Скрипты Обливиона для модмейкеров". Почему "модмейкеров", а не "чайников" (dummies), спросите вы? Но скажите, где гарантия, что автор MSFD GhanBuriGhan не засел за новый учебник - "OSFD"?

Для реализации задуманного была создана команда OSFM team, в которую вошли неравнодушные люди - как обычные пользователи, так и представители различных сайтов, посвященных миру TES. О том, как продвигалась работа, вы можете узнать, посетив сайт "Scripting for Oblivion":
http://theelderscrolls.3dn.ru/
Осенью 2006-го года по различным причинам команда переехала на RUMOR (Russian Morrowind web Ring). Стоит упомянуть, что теперь RUMOR расшифровывается как Russian Morrowind-Oblivion web Ring (Ru-M-O-R):
http://forum.rumor.ru/

Немного о скриптах и проекте.

Для того, чтобы включить скрипты в игру, необходимо использовать специальную программу - Tes 4 Construction Set Oblivion, представляющую собой великолепный полнофункциональный редактор. Именно его использовали разработчики Bethesda Game Studios (Bethesda Softworks) для создания игрового мира.

В основу нашего учебника положена информация, размещенная на The Elder Scrolls Construction Set WIKI (далее для краткости просто WIKI) - специального форума на официальном сайте Bethesda, на котором обычные пользователи всегда могут внести любые изменения в тексты статей, обогащая и дополняя их своим опытом. Так, постепенно, WIKI стал основным источником информации по плагиностроению и работе с конструктором .

Несколько разделов WIKI содержат сведения о скриптах и вопросам, связанных с их написанием, для их последующего включения в собственные модификации основной игры - так называемые плагины. Созданные плагины сохраняются в виде файла с расширением ESP и подключаются к игре с помощью специальной программы-оболочки Oblivion Launcher.

Мы, разумеется, не могли проигнорировать такой мощный источник информации, как WIKI. Поэтому на начальном этапе мы занялись переводами статей, посвященных скриптам и работе с редактором скриптов. Учитывая, что WIKI создается не профессионалами, его нельзя считать свободным от ошибок, как это ни прискорбно. Помимо ошибок, в WIKI встречается не совсем четкое изложение своих мыслей авторами статей, поэтому и перевод оных может быть неоднозначным. До сих пор в WIKI не стандартизирован такой важный вопрос, так синтаксис функций. Именно поэтому при написании учебника многие утверждения мы старались по возможности проверять.

Увы, вы должны понимать, что выловить все ошибки, ляпсусы и "очепятки" практически невозможно. Поэтому заранее просим прощения, если таковые обнаружатся, и будем вам очень благодарны, если вы сообщите нам о них.

Свои замечания и предложения по улучшению учебника вы можете оставить на форуме команды OSFM:
http://forum.rumor.ru/index.php?showforum=138.
Мы обязательно учтем их в новых версиях OSFM.

С другой стороны, если вы являетесь автором интересных скриптов, учебных пособий или статей, раскрывающих сложные аспекты скриптинга, мы с удовольствием включим их в учебник, указав, разумеется, ваше авторство на размещённый материал.

Немного о нашей команде.

Команда разработчиков носит название "OSFM Team".
Члены команды:
1. Garin
2. Platinum
3. ZomBoss (Zomb)
4. Anruin (AD)
5. ForceKeeper
6. Willmore
7. Gwathlobal
8. LiLu (LizardOfOzz)
9. Boblen
10. Eugene
11. Суфир
12. TyRun
13. Gelalhor

Я благодарю всех членов команды за хорошую работу, но хочу также выразить благодарность людям, помогавшим нам в разное время в работе над учебником, а именно: Zig, Vitalka, Когтистый и многие другие.

И в заключение.

Наш учебник, надеюсь, поможет Вам в создании ваших уникальных модов. В какой степени нам это удалось, судить Вам, уважаемые скриптологи и модмейкеры.

По вопросам размещения учебника на ваших сайтах обращайтесь к руководителю проекта, то бишь ко мне.
Все возникающие у вас вопросы, замечания, дополнения и предложения вы можете изложить на форуме проекта:
http://forum.rumor.ru/index.php?showforum=138

С уважением, руководитель проекта Aleksandr Garin.
2007 г.

2. Введение

2.1 Как использовать это руководство.

Мы решились взяться за эту трудную работу, рассчитывая, что учебник будет полезен всем - и новичкам, и опытным скриптологам. В любом случае, поскольку процесс написания скриптов в TES 4 усложнился, а сам язык обогатился новыми конструкциями и функциями, мы надеемся, что наш труд не окажется напрасным и будет вам полезен.

Учебник построен аналогично MSFD – от простого к сложному.
В первом томе после знакомства с основами написания скриптов вам будет предложено написать свой самый первый скрипт, создать свой плагин с ним и включить его в игру. Далее, после более подробного изучения синтаксиса и всех команд, вам сможете написать свой второй скрипт - с ловушкой на шкафу – он более сложный, чем первый, но как учебное пособие гораздо полезнее его.

В первый том мы включили описания всех известных на данный момент функций (всего 359), которые могут использоваться в скриптах Обливиона, описания всех консольных команд и описания всех функций, которые вошли в расширитель скриптов OBSE v0009a. Для вашего удобства описания расположены в алфавитном порядке.
В конце первого тома мы разместили различные приложения и предметный указатель по первому тому со ссылками на страницы с нужной информацией.

2.2 Что такое скрипт?

Скриптовый язык, который использован в игре TES 4 Oblivion, является уникальным, его нельзя использовать нигде, кроме самой игры. Но уникальным его можно назвать и по другим причинам. Скрипты дают нам невиданные возможности по модернизации и изменению игрового мира. Они могут то, чего нельзя сделать никакими другими методами.

В скриптовый язык TES 4 привнесено много нового и это откровенно радует - теперь стали возможны вещи, которые раньше нельзя было реализовать в TES 3.

Сами скрипты представляют собой небольшие текстовые фрагменты с программным кодом, который выполняет в игре определенные функции, изменяя таким образом игровой мир. Скрипты могут “вешаться” на различные объекты, а также выполнять определенные действия в диалогах или отслеживать прогресс игрока в квестах.
Одним из самых больших недостатков скриптового языка является отсутствие возможности определения кода нажатой на клавиатуре клавиши или кнопок мыши. Но и здесь уже появились средства, с помощью которых эта проблема решается легко и просто. Речь идет о расширителе скриптового языка OBSE, который содержит на данный момент около сотни вполне работоспособных и очень полезных функций. Ему мы тоже уделим внимание.

Напомним, что скрипт - это интерпретируемый программный код. Следовательно, он не может выполняться самостоятельно - для его работы необходима другая, скомпилированная программа. В нашем случае это делает “движок” игры, который “зашит” в исполняемый exe-файл - Oblivion.exe.

2.3 Что могут скрипты?

На все, что есть в игре, вы можете тем или иным способом воздействовать с помощью скриптов. Огромное многообразие игрового мира Обливион и богатство его ресурсов – в вашем распоряжении. Игра всегда реагирует на действия игрока. Но с помощью скриптов вы можете изменить реакцию игры на эти действия, например, если игрок намеренно или невзначай осквернил святыню, вы можете вызвать непогоду, превратить день в ночь, вызвать гром и молнию...
Скрипты, как универсальное средство воздействия на геймплей, могут использоваться, например, для создания хитрых ловушек, особо сложных квестов или для воспроизведения нужной анимации. Вы можете заставить любого персонажа игры выполнять совсем несвойственные ему действия, а можете сделать его чрезвычайно “умным”, использовав для этого средства управления искусственным интеллектом. Скрипты помогут вам создать свои собственные уникальные заклинания или зачарованные предметы, которые будут выполнять действия, лежащие за пределами обычных зачарований. В общем, возможности огромны.

2.4 Чего не могут скрипты?

Как бы там ни было, а скриптовый язык имеет и свои ограничения. В нем нет того богатства и универсальности функций, которые есть в “больших” языках программирования, таких как СИ++, Pascal или Delphi. Функции Обливиона специализированы. На данный момент известно 359 функций (одна из них - This, появилась в WIKI совсем недавно), но в игре задействовано только 353 из них. Тем не менее никакой избыточности нет – функции в основном писались и добавлялись в игру по необходимости.

Если вы захотите сделать нечто экстраординарное, что выходит за рамки возможностей языка, то, скорее всего, это у вас не получится - вы не можете изменить скомпилированный файл Olivion.exe, а исходные коды игры по понятным причинам Bethesda не опубликовала. К таким сложностям относится и создание новых функций.
В Паскале, например, вы можете создавать столько функций и процедур, сколько вам нужно. Можно даже сказать, что программирование в Паскале заключается именно в написании подпрограмм, которые затем в тексте основной программы просто вызываются по необходимости. Увы, написать на скриптовом языке новую функцию, используя только редактор, не удастся. Простого пути здесь нет. Но мы не утверждаем, что это невозможно. Примером служит расширитель скриптов OBSE, который будет рассматриваться в первом томе.

Впрочем, если вам действительно нужна какая-нибудь функция, никто вам не запрещает обратиться непосредственно к разработчикам OBSE с просьбой написать и включить ее в релиз следующей версии. Именно таким образом появилось множество новых функций. Такая страничка находится на официальном WIKI.

Как бы там ни было, но существуют обходные пути решения некоторых проблем, различные уловки и трюки, и решить ту или иную задачу можно, использовав какой-нибудь нестандартный подход. Во втором томе мы рассмотрим примеры решения некоторых сложных задач.

Есть и другие ограничения. Скрипты могут изменять свойства предметов, но они не могут создать абсолютно новый предмет непосредственно в игре, если его модель и текстуры не были заложены в игру или плагин изначально. Множество функций, доступных для использования, не перекрывают всех потребностей плагиностроителей. Как бы там ни было, а опытные скриптологи находят всевозможные лазейки и успешно решают поставленные задачи.

Чтобы писать скриптовые программы, вам нужно хорошо знать все особенности, нюансы, возможности и ограничения скриптового языка. А мы постараемся вам в этом помочь.

2.5 Основные отличия в скриптинге TES 3 и TES 4

Основные принципы написания скриптов остались незыблемыми. Однако различий действительно много, поэтому мы приведем только основные.

Первое, что бросается в глаза при взгляде на скрипты Обливиона, это то, что теперь скрипт разделён на исполняемые блоки. В принципе, и в Морровинде делалось то же самое, но в более свободном порядке. Например, если раньше для активации объекта надо было написать:

```
CODE
if ( OnActivate == 1)
…
endif
```

причём это можно было ставить где угодно и на каком угодно уровне условий, то теперь для этого надо объявить отдельный блок:

```
CODE
begin OnActivate
…
end
```

В данном случае применяется тип блока OnActivate. Зачем нужен этот блок? Как известно, в Морровинде все скрипты выполнялись в каждом фрейме (кадре) и предшествовали обновлению изображения на экране. Приходилось принимать специальные меры, что ограничить число выполнения тех или иных операций в тексте скрипта, поскольку это усложняло и без того непростые условия обработки информации и, как следствие, понижало fps.
В приведенном примере блок-тип OnActivate будет выполняться только один раз - в первом же фрейме, в котором выполнилось условие активации, в дальнейшем все, что находится в пределах блока «begin OnActivate … end», выполняться не будет.
В первом томе мы более подробно рассмотрим все 30 разновидностей типов блоков.

Другое явное отличие видно уже по окну редактора – это наличие трёх типов скриптов:
Object script - объектные (локальные) скрипты, "привязанные" к объектам.
Quest script - квестовые скрипты, прикреплены к квестам и выполняются глобально, пока выполняется квест. Эти скрипты доступны из любой локации, где бы игрок ни находился. Глобальных скриптов в том виде, которые известны нам по игре Морровинд, в Обливионе, увы, нет. Но вместо них мы с успехом можем использовать квестовые скрипты. Как это сделать, будет рассмотрено во втором томе учебника в главе «Полезные примеры, решения и трюки».
Magic effect script - Магические скрипты. Скрипты, которые связаны с магическими эффектами и магией.
Важным и полезным нововведением является возможность работы с указателями на объекты. Для этого был введен новый тип переменной – ref (reference).

В связи с введением в игру потребностей в еде и сне была разработана новая система искусственного интеллекта – Radiant AI. Как следствие, это повлекло за собой появление пакетов AI, новой анимации, а также скриптовых функций и консольных команд для управления ими. Поскольку теперь стала необходима и более сложная анимация, возможности управления ею стали гораздо шире. В TES 3, например, все анимационные движения для вида от третьего лица были включены в единый длинный файл base_anim.nif, который содержал 6 с небольшим тысяч фреймов. Множество отдельных движений, которые были включены в него, можно было проигрывать в любой последовательности, используя специальные анимационные функции. Внедрить и проиграть новую анимацию можно было, подключив в конструкторе свой собственный анимационный файл (на самом деле это комплект из трех файлов - *.nif, X*.nif и X*.kf.), однако они могли заменить только девять стандартных анимаций бездействия (idle, idle2...idle9).
В Обливионе уже нет такого длинного файла. Анимация разделена и находится в различных файлах, имеющих формат *.kf новой версии. Теперь можно подключить не один, а множество пользовательских файлов с новой анимацией. К сожалению, Bethesda на момент написания учебника так и не опубликовала новый формат файлов KF. Этим, а также отсутствием официального экспортера/импортера для 3dsMax, и объясняется столь долгое отсутствие плагинов с новой анимацией. Но мир не стоит на месте - усилиями энтузиастов наконец-то появились первые проблески - почти расшифрованы новые

форматы, появились экспортеры/импортеры и появились первые плагины с новой анимацией. Более подробно анимация будет рассмотрена во втором томе.

Изменения коснулись и формата файлов NIF. Все 3d-модели объектов Обливион, имеющие новый формат NIF - v20.0.0.5, не открываются в старом конструкторе и привычных каждому модмейкеру программах. В связи с этим появляются новые версии программ, “понимающих” данный формат. Одной из динамично развивающихся и очень полезных программ является NIFScope. Но основной программой для создания новых объектов является, безусловно, 3dsMax. Все объекты TES 3 Morrowind были разработаны в 3dsMax v4.2. Работа со старшими версиями этой программы была проблематичной. Развитие 3d-технологий привело к тому, что теперь для моделирования объектов Обливиона можно использовать 3ds Max вплоть до 9-й версии. Необходим только соответствующий плагин для вашей версии программы.

Изменения коснулись также и методов управления погодой и освещением. Использование прогрессивной технологии HDR и шейдерных эффектов третьего поколения вызвало необходимость введения для них средств управления (вы наверняка решали вопрос оптимизации настроек для своего компьютера, особенно если он у вас не самый новый или видеокарта не поддерживает “третьи” шейдеры) и новых скриптовых функций.

Об озвучивании уже много писалось. Большой минус локализованной Акеллой и 1С русской версии игры – это отсутствие русской речи. Будем надеяться, что “народные” проекты озвучивания окажутся удачными. Тогда будет иметь смысл делать и свои “озвученные” плагины и использовать для этого средства управления звуком в диалогах. В игре TES 4 Oblivion есть возможность построения осмысленной беседы целой группы персонажей, чего было сложно добиться в TES 3 Morrowind. Во втором томе мы приведем примеры скриптов, показывающих, каким образом это можно реализовать.

Это только малая часть отличий. В общем, читайте учебник, сравнивайте, анализируйте, творите…

3. Обучающий курс

3.1 С чего начинать?

Первое, что вы должны сделать, это скачать TES 4 Construction Set. Тот факт, что конструктор оказался вне игры, плохо, разумеется, но не страшно, если у вас есть Интернет. Его можно скачать как на официальном сайте Bethesda, так и на множестве других сайтов, посвящённых серии игр The Elder Scrolls.
Конструктор прекрасно работает и с русской локализованной версией игры, и с локализованными на русский язык плагинами.

Для людей, плохо знающих английский язык, существует русификатор конструктора. Мы решили помочь автору русификатора Serj777 в переводе некоторых английских терминов. Отчасти такое решение было принято для того, чтобы стандартизировать некоторые термины, которые разные переводчики переводят совершенно по-разному.

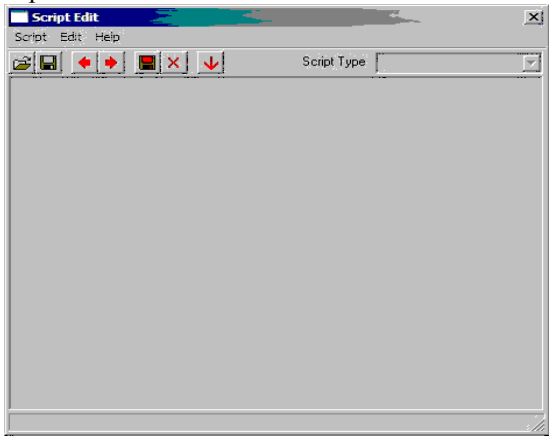
Разумеется, вам нужна и сама игра. Если вы разрабатываете собственные плагины, без лицензионной версии игры от 1С вам не обойтись. Никаких пиратских и полупиратских версий! Есть важные вопросы совместимости и об этом надо помнить. Разумеется, если ваш плагин разрабатывается для англоязычной аудитории, вам придется использовать лицензионную английскую версию игры.

Одним из проектов, который попал в поле зрения нашей команды, стал редактор скриптов TES Script Master (автор Sufir). Он существовал в очень простой версии и мы решили дополнить его различными улучшениями. Он стал более удобен, имеет цветовую подсветку кода, контекстные подсказки и возможность вставки сразу всей конструкции из команд целиком. В будущем, надеемся, TES Script Master будет иметь развитый HELP с полными описаниями всех команд, функций и синтаксиса. Единственное, что не будет сделано – это встроенные компилятор и расширенная проверка синтаксиса.

3.2 Редактор скриптов (Edit Scripts)

3.2.1. Основные сведения.

Редактор скриптов – это простой текстовый редактор, с помощью которого вы можете создавать и редактировать свои скрипты.



В верхней части окна находится главное меню. Пунктов меню всего три – Script (скрипт), Edit (редактировать) и Help (помощь).

3.2.2. Пункт меню Script.
При активизации кнопкой мыши открывается ниспадающее меню, имеющие следующие подпункты:
New (новый скрипт) - создать новый скрипт.
Open (открыть скрипт) - открыть уже существующий скрипт.
Next script (следующий скрипт) - перейти на следующий (по алфавиту) скрипт из списка Open, при этом текущий не сохраняется.
Previous script (предыдущий скрипт) - открыть предыдущий скрипт, при этом текущий не сохраняется;
Save (сохранить) - скомпилировать и сохранить скрипт. Если при компиляции возникают ошибки, то сохранение не производится. Исправьте найденные ошибки и сохраните снова.
Recompile all (скомпилировать всё) - перекомпилировать все скрипты в игре. Используется для проверки и поиска ошибок при работе с большим количеством скриптов (может занять некоторое время). Старайтесь не пользоваться этой опцией. По имеющимся отзывам, в том числе и на WIKI, делать это крайне не рекомендуется!
Delete (удалить) - удалить открытый скрипт.
Exit – закрыть редактор.

3.2.3. Пункт главного меню Edit.
Undo (отменить, Ctrl+Z) - отменяет ваше последнее действие.
Redo (возвратить, Ctrl+Y) - возвращает изменение в скрипте.
Find text (найти текст) - поиск заданного текста в вашем скрипте.
Find next (найти следующее, F3) – поиск, начиная с текущей позиции;
Go to line (перейти на строку) - переход на строку с указанным номером.

3.2.4. Пункт главного меню Help.
При выборе подпунктов этого меню вызывается запуск Интернет-браузера и производится попытка связаться с сервером www.cs.elderscrolls.com, на котором находится WIKI.

3.2.5. Панель инструментов.
Под главным меню находится панель инструментов с иконками, которые дублируют часть пунктов главного меню, а также поле для выбора типа скрипта - Script type. В нем вы указываете, к чему привязывается ваш скрипт – к обьекту (object), квесту (quest) или к магическому эффекту (magic effect).

Иконки по порядку слева направо:
Открытая папка – открыть существующий скрипт.
Дискетка – скомпилировать и сохранить скрипт.
Красная стрелка влево – предыдущий скрипт.
Красная стрелка вправо – следующий скрипт.
Красная дискетка – скомпилировать ВСЕ скрипты. Не поддайтесь соблазну!
Крестик – удалить скрипт.
Стрелка вниз – закрыть.

3.3 Назначение и главная цель вашего скрипта
Приняв решение написать скрипт, вы должны четко представлять себе, какие функции он должен выполнять и нельзя ли обойтись без него. Дело в том, что любой скрипт занимает драгоценные миллисекунды вашего процессорного времени, понижая таким образом fps. Особенно это заметно, когда в сцене находится много персонажей или, например, деревьев. Много скриптов в такой сцене могут резко понизить производительность вашего процессора, сделав из игры “слайд-шоу”. Есть разумный предел, переступать через который нежелательно.

Приняв решение о необходимости данного скрипта и убедившись, что никакими другими более простыми средствами добиться желаемого не удастся, необходимо четко сформулировать задачу с ясными и понятными целями. Когда цель ясна, нужно выбрать оптимальный вариант решения вашей задачи. Основные требования достаточно просты – скрипт должен быть как можно короче и, если нет обязательного выполнения в каждом фрейме, то выбрать оптимальный временной режим выполнения данного скрипта. Иногда задачу можно решить множеством способов. Среди них нужно найти наиболее оптимальный.

3.4 Ваш первый скрипт
Надеемся, вы уже знаете, как запустить редактор TES CS и что там делать, так что перейдём непосредственно к скриптам. В панели инструментов наверху найдите кнопку с карандашом (крайняя справа) – эта кнопка открывает редактор скриптов. Нажали? Значит, вы уже видите этот самый редактор. В нём сейчас пусто и основное его поле, где вам предстоит набирать будущий текст скрипта, имеет серый цвет.

Теперь перейдём к скриптам. Для начала можете посмотреть, как они вообще выглядят. Для этого нажмите на красную стрелку “вправо” и перед вами откроется первый в списке скрипт - AbandonedMineTrap02Script. Судя по названию, он отвечает за работу какой-то ловушки в заброшенной шахте. Всегда старайтесь давать своим скриптам понятные названия!

В этом скрипте есть все основные элементы скрипта:

В самой первой строке пишется название скрипта:
scriptName AbandonedMineTrap02Script – эта строка обязательна, в ней пишется имя скрипта.
Чуть ниже идёт объявление переменных: short triggered – создание короткой целочисленной переменной. В скрипте может не быть вообще переменных, а может быть несколько десятков и разных типов.
Далее идут рабочие блоки:
CODE
begin [тип блока]
….
End

Именно в этих блоках и будет описываться работа скрипта.
А теперь создадим свой собственный скрипт!
В Object Window (окне объектов конструктора) отыщите отмычки. Вы найдете их в списке Items → Misc Item. Вместо отмычки подойдет любой обычный предмет - яблоко или что-нибудь подобное. Сделайте двойной щелчок на предмете. Откроется диалоговое окно со свойствами предмета. Там, где написано "Script: NONE", кликните на кнопку с тремя точками, находящуюся справа. Это откроет редактор скриптов.
Выберите Script → New, чтобы создать новый скрипт.
Скопируйте приведенный ниже программный код комбинацией клавиш Ctrl + C и вставьте его в окно редактора с помощью комбинации Ctrl+V:
CODE
ScriptName HelloWorld
Begin OnAdd
Message "Hello World!", 10
end
Информация о данном скрипте:
Первая строка – это имя скрипта (Scriptname). Убедитесь, что оно уникально. В противном случае вам придётся его изменить, т.к. редактор не позволит скомпилировать скрипт.
"Begin OnAdd" означает, что код внутри данного блока будет исполняться всякий раз, когда вы подберете нужный предмет.
"OnAdd" – это тип данного блока. Команда “end” закрывает секцию скрипта, которая связана с “OnAdd”. У вас может быть несколько отдельных блоков begin/end, имеющие одинаковые или различные типы блоков (blocktype).
Между begin и end находится исполняемый программный код. В нашем случае это всего лишь одна функция – Message, которая выведет на экран сообщение "Hello World!".
И последнее – временной контроль за строкой сообщения. В данном случае она имеет параметр «10», который означает, что строка будет отображаться на экране в течение 10 сек.
Сохраните скрипт и закройте Редактор скриптов.
Вернитесь к свойствам предмета, "HelloWorld" должно появиться как опция в ниспадающем списке скриптов. (Возможно, вам нужно будет закрыть окно свойств объекта и снова открыть его, чтобы скрипт "HelloWorld" появился в ниспадающем списке.)
Щелкните ОК, чтобы сохранить изменения, и закройте окно.
На панели инструментов выберите File -> Save; в окне сохранения введите имя файла HelloWorld (это будет ваш первый *.esp файл плагина - HelloWorld.esp !).
Откройте Oblivion_Launcher и щелкните Data Files.
Двойным щелчком мыши подключите свой HelloWorld.esp, щелкните OK и закройте окно.
Теперь все должно работать! Запустите игру, бросьте отмычку (или предмет, который вы использовали) на землю и подберите ее. При этом должно появиться сообщение "Hello World!".
И с этим первым результатом добро пожаловать в скриптинг!

4. Основы скриптинга

4.1 Общая информация

Скрипты – это маленькие программы, работающие непосредственно в игре. Скриптовый язык Обливиона довольно прост, и ваш опыт в любом другом языке программирования будет весьма полезен.

Основное правило при написании скриптов: каждая операция должна записываться в новой строке.
Вы можете также снабдить ваш код комментариями, используя ";". Все, что стоит после точки с запятой, будет в дальнейшем игнорироваться компилятором и игрой, но на этапе разработки и обсуждения скрипта комментарии облегчат процесс его понимания.

Скриптовые операции делятся на две основные категории, каждая из которых имеет две подкатегории:
Команды: административные и утверждения.
Функции: активные и пассивные.

Команды

Команда – это скриптовая операция, контролирующая все, что происходит в скрипте. Команды регулируют выполнение скрипта, но не влияют на игровой мир непосредственно. Команды делятся на две подкатегории:

Административные команды – это Scriptname, Begin...End и Blocktypes (типы блоков), используемые командой Begin. В каждом скрипте должны быть по крайней мере команда Scriptname, и один Begin...End блок - для назначения скрипта на объекты и определения, когда тот должен заработать.
Утверждения – это Set, If, Return и команды объявления переменных (variable declarations). Утверждения не запускают скрипт, но во всех скриптах, кроме самых простых, используются для управления их работой.
Функции

Функции – это скриптовые операции, которые взаимодействуют с игровым миром. Это самая большая категория скриптовых операций. Каждая функция возвращает значение, которое может быть использовано при проверке условий "if" или которое можно сохранить в переменной, используя команду set.

Функции подразделяются на две подкатегории: пассивные и активные:
Пассивные функции проверяют определённые игровые величины и возвращают их значения. Например, функция GetActorValue возвращает определённое значение характеристики актера, а функция GetDetected - только проверяет и возвращает "1", если цель обнаружена, или "0", если нет.
Активные функции вносят изменения в игровой мир и обычно возвращают логический результат ("1" или "0") - были ли действия успешными или нет. RemoveSpell, к примеру, снимает заклинание с цели и возвращает "1", если действие прошло успешно (т.к. на цели было заклятие). PlaceAtME создает какой-либо объект в локации вызвавшего эту функцию и возвращает ссылку на копию этого объекта.
Ссылки и переменные

Функции оказывают действие на "вызывающий объект" (тот, на котором работает скрипт) по умолчанию, но вы можете вызвать их и на другом объекте, используя ссылку на объект. Когда вы хотите, чтобы скрипт влиял, например, на игрока, используйте ссылку на игрока (“Player”) и точку “.”, чтобы перенаправить вызов функции (function-call) на игрока с вызывающего объекта:
CODE
player.additem gold_001 100

Для большей гибкости вместо числовых значений вы можете использовать переменные, если они требуются для какой-нибудь функции:
CODE
short addgold
set addgold to 100
player.additem gold_001 addgold

Для своих целей вы можете использовать как локальные переменные, объявленные в этом же скрипте, так и глобальные. Из функции нельзя напрямую обратиться к переменной в другом скрипте. Но такой возможностью обладают команды If и Set.
Например, эта строка скрипта работать не будет:
CODE
player.additem gold_001 otherobject.addgold

Следует написать так:
CODE
short addgold ; объявление вспомогательной локальной переменной
set addgold to otherobject.addgold ; присвоить ей значение внешней локальной переменной
player.additem gold_001 addgold ; использование полученных внешних данных

4.2 Типы скриптов

4.2.1 Общие сведения о типах скриптов
Открывая меню свойств объекта, вы, вероятно, заметили, что в нём есть поле для подключения скриптовой программы (скрипта). Скрипт - это, как правило, небольшая программа, которая может быть присвоена объекту игрового мира, что позволяет ему выполнять некие особенные действия.
Скрипты можно условно разделить на две главные категории:
Локальные (Reference scripts) - скрипты на объектах игрового мира.
Нелокальные скрипты (Non-reference scripts).
Локальные скрипты.

Локальные скрипты (Reference scripts) - это скрипты, которые запускаются на объектах. Они разделяются на два типа:
Скрипты, прикрепленные к объектам - объектные скрипты (References object scripts)
Скрипты в поле "result" диалогов (dialogue results)
В таких скриптах некоторые локальные функции (Reference functions) могут использоваться без явного указания ID объекта (ID_NameObject) – в этом случае они будут нацелены на объект, с которым этот скрипт связан (с объектом, на который он “повешен”).
Другими словами, локальные скрипты – это скрипты, в которых функции могут использовать неявные обращения к объекту, к которому они прикреплены. Например, если к NPC прикреплен локальный скрипт, то функция additem в этом скрипте добавит ему 100 золотых:

CODE
additem gold_001 100

Нелокальные скрипты.
В нелокальных скриптах (квестовые скрипты, результаты выполнения стадии квеста и пр.) имена объектов (ID_NameObject) в функциях должны быть указаны явно. Другими словами, нелокальные скрипты – это скрипты, в которых функции должны явно указывать объект, в отношении которого они вызываются, потому что они не привязаны к какой-либо копии объекта:
Квестовые скрипты – скрипт можно сделать квестовым, если указать его тип "Quest" в окне редактирования скриптов.
Скрипты в поле результата стадии квеста.

С другой стороны, все скрипты можно разделить на:
Именованные скрипты (Named Scripts)
Результирующие (Result scripts).

Именованные скрипты.
Именованные скрипты – это полные скрипты, которые могут быть связаны с объектами, квестами или магическими эффектами и которые могут использовать переменные и begin/end блоки.
Именованные скрипты создаются с использованием окна редактирования скриптов. Эти скрипты могут в полной мере использовать все возможности скриптового языка. Командой scriptname им обязательно нужно дать имя.
Именованные скрипты могут быть трех типов:
Скрипты на объектах (объектные скрипты)
Квестовые скрипты
Скрипты для магических эффектов
и могут быть присоединены к объектам, квестам или магическим эффектам соответственно. Когда заскриптованный объект помещается в игровой мир (то есть, создается новая копия объекта), этот объект будет иметь собственные копии переменных, объявленных в скрипте. Таким образом, один и тот же скрипт может быть использован для контроля за состоянием нескольких копий объекта или даже к различным объектам.
Объектные скрипты. Скрипты, “повешенные” на объектах - это именованные скрипты, которые могут быть прикреплены к любому объекту. Это всегда локальные скрипты, и они могут использовать любые функции, кроме специализированных команд для магических эффектов.

Квестовые скрипты создаются с типом скрипта Quest, выбранном в окне редактирования скриптов. Никакой другой тип скрипта работать с квестами не будет. Квестовые скрипты - это нелокальные скрипты и потому должны использовать особый синтаксис в вызове функций.

Основные положения:
Квестовые скрипты выполняются только тогда, когда выполняется квест (вы можете определить это в игре, напечатав в консоли sqv QUEST_NAME). Квесты начинаются и заканчиваются с помощью команд StartQuest и StopQuest. Эти команды не зависят от того, завершен квест или нет. Завершение квеста означает, что он перемещается на панель завершенных квестов (Completed Quest tab) в журнале игрока – но скрипт все равно будет исполняться, пока квест не будет переведен в пассивное состояние командой StopQuest.
Квест автоматически начинается, когда в журнал игрока записывается запись от этого квеста. Поэтому команда SetStage QUESTNAME 10 (если запись 10 содержит текст) автоматически начнет квест.
В общем, вы должны останавливать квест, когда он завершен, чтобы скрипт, прикрепленный к квесту, перестал исполняться. Если по какой-то причине квест должен оставаться активным (у вас есть диалог или скрипт, необходимый после окончания квеста), создайте второй квест (например, MS38 и MS38FIN).
Значения квестовых переменных можно запрашивать и изменять, даже когда квест не активен. Когда квест останавливается, его скрипт перестает исполняться, но он все равно существует и его переменные также остаются в целости.
Как изменить частоту выполнения квестового скрипта:
Объявите "магическую переменную" float fQuestDelayTime в вашем квестовом скрипте.
Присвойте переменной fQuestDelayTime число, определяющее требуемое время между циклами выполнения скрипта, в секундах. Чем меньше это значение, тем чаще будет выполняться ваш скрипт. Если число будет очень мало, например, 0.01 сек, то это будет соответствовать частоте исполнения скрипта 100 раз в секунду. Поскольку частота смены кадров (фреймов) на экране (fps) намного меньше, то ваш скрипт будет исполняться в каждом фрейме.
Если вы установите его в 0, скрипт будет выполняться каждые 5 секунд (время по умолчанию). Используйте эту возможность с осторожностью и в особых случаях – значение fQuestDelayTime должно быть меньше 5-ти.
Скрипты для магических эффектов – это особые скрипты, которые можно использовать для создания заскриптованных магических эффектов. Они создаются в окне редактирования скриптов, но у них должен быть тип Magic Effect, иначе использовать магические эффекты не удастся. Такой скрипт будет выполняться, если игрок находится рядом, и будет останавливаться вместе с игрой при вызове какого-либо меню.

Эти скрипты не используют обычные типы блоков begin/end, поскольку выполняются, пока эффект заклинания активен на цели. Для них существуют три специальных блока и одна особая функция.

Типы блоков для магических эффектов:
ScriptEffectStart
ScriptEffectFinish

ScriptEffectUpdate
Другие типы блоков использовать нельзя (они будут скомпилированы, но код внутри них выполняться не будет).
Специальная функция для магических эффектов:
ScriptEffectElapsedSeconds
В остальном, скрипты для магических эффектов ведут себя как обычные локальные скрипты и внутри блоков для магических эффектов можно использовать обычные скриптовые функции.

Результирующие скрипты.
Результирующие скрипты (Result scripts) – фрагменты скриптов, которые связаны с диалогами или стадиями квестов и возвращающие некий результат в поле result диалогового окна.
Скрипты в поле “result” (Result scripts) диалоговых окон конструктора – это скриптовые фрагменты, которые выполняются один раз, когда наступает определенное событие. Существует два типа таких скриптов:
Скрипты в поле result диалога выполняются, когда NPC говорит связанное со скриптом сообщение. Такие скрипты, как и скрипты, прикрепленные к объектам, являются локальными скриптами.
Скрипты в поле result стадии квеста выполняются, когда достигнута определенная стадия квеста, и НУЖНЫЙ ПАРАМЕТР предмета или сам предмет стадии квеста (stage item) соответствует заданным условиям. Эти скрипты относятся к нелокальным скриптам.
Скрипты в поле result имеют следующие ограничения по сравнению с именованными скриптами:
В них нельзя объявлять переменные.
В них нельзя использовать блоки begin/end.
В диалоговых скриптах нельзя использовать неявное обращение к переменным локального скрипта NPC.
В остальном, скрипты в поле result могут использовать все обычные функции и команды.

4.2.2 Как часто обрабатываются скрипты (Script Processing)
Когда и каким образом скрипты обрабатываются, зависит от того, с чем связан скрипт. В общем случае локальные скрипты на объектах выполняются только тогда, когда игрок находится рядом; квестовые же скрипты выполняются постоянно - до тех пор, пока выполняется квест.
Как часто выполняются скрипты:
Квестовые скрипты: когда квест активен, выполняются каждые 5 секунд (по умолчанию). Вы можете изменять частоту выполнения скрипта, изменив переменную float fQuestDelayTime в теле скрипта.
Скрипты на персонажах (существах и NPC): выполняются каждый раз, когда выполняется пакет ИИ (AI) актера. При наивысшем приоритете (загруженная область вокруг игрока) они выполняются в каждом фрейме. Если же игрока рядом нет, то гораздо реже (вплоть до одного раза за 15 минут игрового времени при самом низком приоритете). Но это единственные скрипты (кроме квестовых скриптов), которые выполняются, когда игрока нет рядом.
Скрипты на копиях: выполняются в каждом фрейме, когда ячейка загружена, совсем не выполняются, когда ячейка не загружена. То есть, они выполняются, только когда игрок рядом (что означает, что это хороший вариант для использования ресурсоемких скриптов, проверяющих, например, дистанцию).
Скрипты на объектах в контейнерах: выполняются, когда выполняются скрипты на контейнерах – поэтому скрипты вещей у актеров выполняются, когда обрабатывается актер; вещи в других контейнерах обрабатываются в каждом фрейме, когда ячейка загружена.
Скрипты на дверях: особый случай - эти скрипты выполняются, как скрипты на копиях (в каждом фрейме в загруженной ячейке), но они также выполняются один раз, когда персонаж взаимодействует с дверью.

4.3 Команды (Commands)

4.3.1 Общие сведения о командах (Commands)
Скриптовый язык Обливиона специально создан для игры, но тем не менее структурой он очень напоминает другие языки программирования. Самый простой скрипт имеет три элемента: команда ScriptName (название скрипта), команда begin с указанием типа блока и команда end. Пример:
CODE
ScriptName MyScript
begin OnAdd Player
; делать что-то
End

Команда ScriptName понятна: она задает имя скрипта. Она обязательна и имя должно быть уникальным.
Команда begin определяет действие, которое запускает скрипт. Если условие, указанное в команде begin (тип блока), выполняется, то исполняются все команды, находящиеся в пределах блока между begin и end.

Каждая команда begin обязательно должна иметь соответствующую ей команду end!

Вы можете прервать выполнение своего скрипта командой return. Это может оказаться полезно внутри команды if в случаях, когда вам не нужно продолжать выполнение скрипта дальше.
Вы можете добавить комментарий к какой-нибудь строке скрипта, используя точку с запятой. Все, что стоит в строке после них, игнорируется компилятором, и это позволяет вам делать заметки по поводу того, что делает этот скрипт.

4.3.2 Команда Scriptname (название скрипта)

Все скрипты (точнее, именованные скрипты) должны начинаться с этой команды, которая задает имя скрипта. Имя должно быть уникальным и состоять из одного слова.
CODE
scriptname TestScript

или
CODE
scn Testscript

4.3.3 Команды Declaring variables (команды объявления переменных)

Существует 4 типа переменных:
Short - Короткая целочисленная переменная.
Long - Длинная целочисленная переменная.
Float - Вещественная переменная.
Ref - Переменная типа reference (ссылка на копию объекта).
CODE
short от -32,768 до 32,767
long от - 2,147,483,648 до 2,147,483,647
float от -3.402823×1038 до -1.175494×10−38, 0 и от 1.175494×10−38 до 3.402823×1038 (с точностью до 7 знаков)
ref 32-битный код FormID

Первые три типа переменных можно объявлять и как локальные, и как глобальные. Имена переменных не чувствительны к регистру. Объявление происходит вместе с типом и именем:
CODE
short myShortVariable
long myLongVariable
float myFloatVariable

Локальная переменная может быть объявлена где угодно в скрипте, главное, чтобы она была объявлена перед первой командой, которая будет ее использовать. Обычно все переменные декларируют в начале скрипта, чтобы его было легче читать.

Скрипты также могут использовать переменные типа reference (ссылка на копию объекта):
CODE
ref myRefVariable

4.3.4 Команды Begin (начало блока) и End (конец блока)
Конструкции Begin-End являются блоками, внутри которых и размещен, собственно, весь исполняемый код скрипта. Другими словами, все скриптовые команды, кроме объявления переменных, обязательно должны находиться именно внутри блоков begin-end!

Теперь все блоки Begin-End имеют свое условие для выполнения. Это полезное нововведение в скриптовый язык TES 4 Oblivion, по сравнению с TES 3 Morrowind. Условия ставятся в строке непосредственно после команды Begin и получили название "тип блока" (BlockTypes). Блок-типы имеют различное назначение и представляют собой некое условие. Каждый раз, когда скрипт выполняется в текущем фрейме, все блоки скрипта проверяются на истинность их условий. Если условие какого либо блока не истинно, то этот блок и все скриптовые команды в его пределах выполняться не будут.

Команда End всегда завершает начатый блок, т.е. каждому begin должен обязательно соответствовать свой end.

Количество блоков в скрипте может быть произвольным, но не менее одного!
Блоки следуют один за другим последовательно.
Важное замечание: блоки begin-end не могут быть вложенными!
Все блоки должны обязательно содержать название используемого типа блока!

Пример:
CODE
begin GameMode
end

В таблицу, приведенную ниже, сведены все существующие блок-типы. Общее их количество равно 30.
CODE

N	Тип блока	Параметры	Описание

1 | GameMode | нет | Исполняется в каждом фрейме, когда игра не находится в меню.
| | | Большинство скриптов используют исключительно этот тип блока.
2 | MenuMode | Тип меню (не обязательно)
| | | Исполняется в каждом фрейме, пока игра находится в меню.
3 | OnActivate | нет | Исполняется один раз, когда объект активирован.
4 | OnActorEquip | ID объекта
| | | Исполняется один раз, когда заскриптованный актер надевает
| | | указанный объект.
5 | OnActorUnequip | ID объекта
| | | Исполняется один раз, когда заскриптованный актер снимает
| | | указанный объект.
6 | OnAdd | ID копии контейнера (не обязательно)
| | | Исполняется один раз, когда объект добавляется в инвентарь
| | | контейнера.
7 | OnAlarm | Тип преступления, Преступник (не обязательно)
| | | Исполняется один раз, когда актер поднимает тревогу по
| | | поводу указанного преступления, совершенного преступником
| | | (актером).
8 | OnAlarmVictim | Тип преступления, Жертва (не обязательно)
| | | Исполняется один раз, когда актер поднимает тревогу
| | | по поводу указанного преступления, совершенного против
| | | жертвы (актером).
9 | OnDeath | ID актера (не обязательно)
| | | Исполняется один раз, когда умирает от рук указанного
| | | актера.
10 | OnDrop | ID копии контейнера (не обязательно)
| | | Исполняется один раз, когда объект сброшен из контейнера.
11 | OnEquip | ID актера (не обязательно)
| | | Исполняется один раз, когда объект надевается указанным
| | | актером.
12 | OnHit | ID актера (не обязательно)
| | | Исполняется один раз, когда получают удар от указанного
| | | актера
13 | OnHitWith | ID объекта (не обязательно)
| | | Исполняется один раз, когда актер получает удар указанным
| | | оружием
14 | OnKnockout | нет | Исполняется один раз, когда отправлен в нокаут
| | | указанными актером
15 | OnLoad | нет | Исполняется один раз, когда в игру загружается модель
| | | (3D) объекта
16 | OnMagicEffectHit | ID эффекта (не обязательно)
| | | Исполняется один раз, когда на актера накладывается
| | | указанный магический эффект
17 | OnMurder | ID актера (не обязательно)
| | | Исполняется один раз, когда указанный актер совершает
| | | убийство актера (то есть преступление)
18 | OnPackageChange | ID пакет | Исполняется один раз, когда актер меняет указанный
| | | пакет ИИ
19 | OnPackageDone | ID пакета | Исполняется один раз, когда актер завершает
| | | указанный пакет ИИ
20 | OnPackageStart | ID пакета | Исполняется один раз, когда актер начинает
| | | указанный пакет ИИ
21 | OnReset | нет | Исполняется один раз, когда ячейка с заскриптованным
| | | объектом сбрасывается (reset)
22 | OnSell | ID копии продавца (не обязательно)
| | | Исполняется один раз, когда объект продается указанным
| | | продавцом
23 | OnStartCombat | ID актера-цели (не обязательно)
| | | Исполняется один раз, когда актер начинает битву с указанным
| | | актером
24 | OnTrigger | ID копии сталкивающегося объекта (не обязательно)
| | | Исполняется один раз, когда объект сталкивается с указанным
| | | объектом
25 | OnTriggerActor | ID копии сталкивающегося объекта (не обязательно)
| | | Исполняется один раз, когда объект сталкивается с указанным
| | | актером
26 | OnTriggerMob | ID копии сталкивающегося объекта (не обязательно)
| | | Исполняется один раз, когда объект сталкивается с указанным

| | | мобильным объектом (актеры, стрелы, магические снаряды)
27 | OnUnequip | ID копии контейнера (не обязательно)
| | | Исполняется один раз, когда снимается указанным актером.
28 | ScriptEffectStart | нет | Особый тип блока, используется только в скриптах
| | | магических эффектов
29 | ScriptEffectFinish| нет | Особый тип блока, используется только в скриптах
| | | магических эффектов
30 | ScriptEffectUpdate| нет | Особый тип блока, используется только в скриптах
| | | магических эффектов

Заметьте, что для блоков с параметрами можно указывать тот же самый блок несколько раз, используя разные параметры. Например, этот скрипт не содержит ошибок:
CODE
begin OnAdd
; какой-то скрипт выполняется каждый раз, когда этот объект добавляется кому-то в инвентарь
end
begin OnAdd player
; Какой-то скрипт выполняется каждый раз, когда этот объект добавляется в инвентарь игрока.
; Заметьте, что блок OnAdd без параметров ТАКЖЕ будет исполнен.
end
begin OnAdd MysteriousChest
; Какой-то скрипт выполняется каждый раз, когда этот объект
; добавляется в инвентарь контейнера MesteriousChest.
; Заметьте, что блок OnAdd без параметров ТАКЖЕ будет исполнен.
End

4.3.5 Типы выполняемых блоков (BlockTypes).
Рассмотрим типы выполняемых блоков более подробно.

GameMode

Скрипт внутри этого блока выполняется в каждом фрейме, когда игра не находится в меню. Большинство скриптов используют исключительно этот тип блока.
Пример:
CODE
; Пример скрипта-таймера
scn myScript

float timer
short init

begin GameMode
if init == 0
; установить значение таймера
set timer to 25
set init to 1
else
if timer > 0
set timer to timer - getSecondsPassed
else
; здесь код для того, что должно произойти через 25 секунд
endif
endif
end

В игре Обливион блок-тип GameMode используется 964 раза.
Примеры скриптов: AbandonedMineTrap02Script, AloysBincalScript, AltarofAkatosh

MenuMode

Синтаксис:
CODE
begin MenuMode MenuType (не обязательно)

Пример:
CODE
begin MenuMode

begin MenuMode 1

Без параметров этот блок будет выполняться, когда игрок НЕ находится режиме игры – то есть, когда отображается ЛЮБОЕ меню.
Если вы включите параметр, то можете указать тип меню, или конкретное меню, во время показа которого должен выполняться блок:

Тип меню:
1 = "четыре главных" (интерфейс персонажа: хар-ки, магия, инвентарь, журнал)
2 = любые другие меню (окна сообщений, контейнеры и т.д.)
3 = консоль
Меню:
1001 = Сообщение
1002 = Инвентарь
1003 = Хар-ки
1004 = Главный HUD (HUDMain)
1005 = Информационный HUD (HUDInfo)
1006 = Крест (HUDReticle)
1007 = Заставка в время загрузки
1008 = Контейнер
1009 = Диалог
1010 = Субтитры (HUDSubtitle)
1011 = Общий (Generic)
1012 = Спать/Ждать
1013 = Пауза
1014 = Взлом
1015 = Опции
1016 = Количество
1017 = Аудио
1018 = Видео
1019 = Видеодисплей (VideoDisplay)
1020 = Геймплей (Gameplay)
1021 = Управление (Controls)
1022 = Магия
1023 = Карта
1024 = Магическое всплывающее окно (MagicPopup)
1025 = Торговля (Negotiate)
1026 = Книга
1027 = Уровень
1028 = Тренировка
1029 = Знак
1030 = Класс
1031 = Атрибуты
1032 = Умения
1033 = Специализация
1034 = Убеждения
1035 = Починка
1036 = Раса/Пол
1037 = Покупка заклинаний
1038 = Загрузка
1039 = Сохранение
1040 = Алхимия
1041 = Создание заклинаний
1042 = Зачарование
1043 = Установка эффектов
1044 = Главное
1045 = Воздух
1046 = Быстрые клавиши
1047 = Разработчики (Credits)
1048 = Камень Сигил
1049 = Перезарядка
1051 = Редактирование текста

В игре Обливион блок-тип MenuMode используется 34 раза.
Примеры скриптов: BedDiseaseSCRIPT, Blade3Script, DAHermaeusScript

OnActivate

Синтаксис:
CODE
begin OnActivate

Этот блок запускается один раз, когда активируется заскриптованный объект.
Активирование объекта устанавливает "бит активирования". Поэтому многократные последовательные вызовы Activate на одном объекте приведут лишь к однократному проходу через блок OnActivate.
Заметьте, что это блокирует нормальное активирование объекта. Чтобы использовать активирование объекта по умолчанию, нужно вызвать на нем Activate. Если вы хотите сделать что-то особенное в зависимости от того, что активировало объект, используйте IsActionRef внутри блока OnActivate.
Помните, что у активатора НЕТ действия по умолчанию, которое выполняется при активации.

В игре Обливион блок-тип OnActivate используется 587 раз.
Примеры скриптов: ActRockGreatForest01SCRIPT, AltarofAkatosh, AltarofArkay

OnActorEquip

Синтаксис:
CODE
begin OnActorEquip ObjectID

Пример:
CODE
begin OnActorEquip DrinkMead

Этот блок выполняется один раз, когда заскриптованный актер надевает указанный объект (ObjectID).

В игре Обливион блок-тип OnActorEquip используется 1 раз.
Примеры скриптов: Dark14AlvalUvaniScript

OnActorUnequip

Синтаксис:
CODE
begin OnActorUnequip ObjectID

Пример:
CODE
begin OnActorUnequip DrinkMead

Этот блок выполняется один раз, когда заскриптованный актер снимает указанный объект (ObjectID).

В игре Обливион блок-тип OnActorUnequip не используется.

OnAdd

Синтаксис:
CODE
begin OnAdd ContainerRefID (не обязательно)

Пример:
CODE
begin OnAdd
begin OnAdd player

Этот блок выполняется один раз, когда заскриптованный объект добавляется в указанный контейнер (ContainerRefID). Если контейнер не указан, блок будет исполняться, если объект добавляется в любой инвентарь.

В игре Обливион блок-тип OnAdd используется 72 раза.
Примеры скриптов: CGAkaviriLongswordScript, CGBladesEquipmentScript, CGBowScript

OnAlarm

Синтаксис:
CODE
begin OnAlarm CrimeType, Criminal (не обязательно)

Пример:

CODE
begin OnAlarm 0
begin OnAlarm 3, player

Этот блок выполняется один раз, когда заскриптованный актер поднимает тревогу по поводу определенного типа преступлений (CrimeType), совершенного преступником-актером (Criminal). Если указан только тип преступлений, блок будет исполняться, когда актер поднимает тревогу по поводу этого типа преступлений (неважно кем совершенного). Если необходимо знать больше информации о преступлении, функция GetCrimeKnown может использоваться внутри блока OnAlarm, чтобы определить, случилась ли определенная комбинация преступник/жертва.

В игре Обливион блок-тип OnAlarm используется 15 раз.
Примеры скриптов: Dark08AlarmScript, Dark08NelsScript, Dark08NevilleScript

OnAlarmVictim

Синтаксис:
CODE
begin OnAlarmVictim CrimeType, Victim (не обязательно)

Пример:
CODE
begin OnAlarmVictim 0
begin OnAlarmVictim 3, player

Этот блок выполняется один раз, когда заскриптованный актер поднимает тревогу по поводу определенного типа преступлений (CrimeType), совершенного против жертвы-актера (Victim). Если указан только тип преступлений, блок будет выполняться, когда актер поднимает тревогу по поводу этого типа преступлений (неважно против кого совершенного). Если необходимо знать больше информации о преступлении, функция GetCrimeKnown может использоваться внутри блока OnAlarm, чтобы определить, случилась ли определенная комбинация преступник/жертва.

В игре Обливион блок-тип OnAlarmVictim не используется.

OnDeath

Синтаксис:
CODE
begin OnDeath ActorID (не обязательно)

Пример:
CODE
begin OnDeath SuperChampion

Этот блок выполняется один раз, когда указанный актер (ActorID) убивает заскриптованного актера. Если параметр не используется, блок выполняется, когда заскриптованный актер умирает.

В игре Обливион блок-тип OnDeath используется 303 раза.
Примеры скриптов: AlawenScript, AlessiaCaroScript, AlixLencoliaScript

OnDrop

Синтаксис:
CODE
begin OnDrop ContainerRefID (не обязательно)

Пример:
CODE
begin OnDrop
begin OnDrop player

Этот блок выполняется один раз, когда заскриптованный объект сбрасывается из указанного контейнера (ContainerRefID). Если контейнер не указан, блок будет исполняться, если объект сбрасывается из любого инвентаря.

В игре Обливион блок-тип OnDrop используется 7 раз.
Примеры скриптов: DarkScalesScript, GoblinHeadScript, HrormirsIcestaffScript

OnEquip

Синтаксис:
CODE
begin OnEquip ActorID (не обязательно)

Пример:
CODE
begin OnEquip
begin OnEquip player

Этот блок исполняется один раз, когда заскриптованный объект надевается указанным актером (ActorID). Если параметр не используется, блок будет исполняться, когда объект надевается любым актером.

В игре Обливион блок-тип OnEquip используется 27 раз.
Примеры скриптов: AmuletofKingsSCRIPT, ArenaRaimentScript, BoarMeatScript

OnHit

Синтаксис:
CODE
begin OnHit ActorID (не обязательно)

Пример:
CODE
begin OnHit BaurusRef

Этот блок исполняется один раз, когда заскриптованный актер получает удар от указанного актера (ActorID) оружием или заклинанием. Если параметр не используется, блок исполняется каждый раз, когда актера ударяют.

В игре Обливион блок-тип OnHit используется 55 раз.
Примеры скриптов: ArenaCombatant, ArenaCombatantBlue, ArenaCombatantMulti

OnHitWith

Синтаксис:
CODE
begin OnHitWith ObjectID (не обязательно)

Пример:
CODE
begin OnHitWith SuperWeapon

Этот блок исполняется один раз, когда заскриптованный актер получает удар указанным оружием (ObjectID). Если параметр не используется, блок будет выполняться, когда заскриптованный актер получает удар любым оружием.

В игре Обливион блок-тип OnHitWith используется 7 раз.
Примеры скриптов: CGRopeBucketScript, Dark05MotierreScript, Dark09AdamusScript

OnKnockout

Синтаксис:
CODE
begin OnKnockout

Этот блок исполняется один раз, когда заскриптованный актер отправляется в нокаут.
В игре Обливион блок-тип OnKnockout не используется.

OnLoad

Синтаксис:
CODE
begin OnLoad

Этот блок исполняется один раз, когда загружается модель объекта, то есть когда игрок заходит в интерьер, ячейка с объектом загружается в область 5x5 вокруг игрока.

В игре Обливион блок-тип OnLoad используется 95 раз.
Примеры скриптов: ARGateAUTOCLOSE01SCRIPT, ARTallWallCarvingAnim01SCRIPT

OnMagicEffectHit

Синтаксис:
CODE
begin OnMagicEffectHit EffectID

Пример:
CODE
begin OnMagicEffectHit FIDG

Этот блок исполняется один раз, когда указанный тип магического эффекта (EffectID) ударяет по заскриптованному объекту. Если эффект не указан, блок будет выполняться, когда любой эффект ударяет по объекту.

В игре Обливион блок-тип OnMagicEffectHit используется 7 раз.
Примеры скриптов: JskarScript, MG05RockScript, MG10ColumnScript

OnMurder

Синтаксис:
CODE
begin OnMurder ActorID (не обязательно)

Пример:
CODE
begin OnMurder SuperChampion

Этот блок выполняется один раз, когда указанный актер (ActorID) совершает убийство (преступление) заскриптованного актера. Если параметр не используется, блок выполняется, когда совершается убийство заскриптованного актера (неважно кем).

В игре Обливион блок-тип OnMurder используется 10 раз.
Примеры скриптов: DarkExiledScript, DarkWrathofSithis2Script, DarkWrathofSithisScript

OnPackageChange

Синтаксис:
CODE
begin OnPackageChange PackageID

Пример:
CODE
begin OnPackageChange FollowPlayerPackage

Этот блок выполняется один раз, когда заскриптованный актер меняет пакет ИИ с указанного (PackageID) на другой. Заметьте, что "прерывающие" пакеты, такие, как бой или разговор, не запускают этот блок, поскольку они лишь временно заменяют текущий пакет.
Блок запускается, когда актер меняет пакет ИИ на другой. Если указанный пакет завершается, но актер снова начинает его исполнять, OnPackageChange НЕ запустится.

В игре Обливион блок-тип OnPackageChange используется 32 раза.
Примеры скриптов: AlessiaCaroScript, BarthelGernandScript, BaurusScript

OnPackageDone

Синтаксис:
CODE
begin OnPackageDone PackageID

Пример:
CODE
begin OnPackageDone FollowPlayerPackage

Этот блок исполняется один раз, когда заскриптованный актер завершает указанный пакет ИИ (PackageID). Заметьте, что пакет может быть завершен из-за неудачи (не найден путь, невозможно найти нужное количество объектов, или истекло время), если только у пакета не установлены флаги Must Reach Location или Must Complete.
Заметьте, что некоторые типы пакетов, например прогулка, сон и еда, не могут «завершиться», поскольку у них нет конца, поэтому OnPackageDone не будет запускаться для этих типов пакетов.
(Блок OnPackageEnd взаимозаменяем с OnPackageDone)

В игре Обливион блок-тип OnPackageDone используется 45 раз.
Примеры скриптов: BaurusScript, BurdSCRIPT, CGEmperorScript

OnPackageStart

Синтаксис:
CODE
begin OnPackageStart PackageID

Пример:
CODE
begin OnPackageStart FollowPlayerPackage

Этот блок исполняется один раз, когда заскриптованный актер начинает выполнять указанный пакет (PackageID).

В игре Обливион блок-тип OnPackageStart используется 13 раз.
Примеры скриптов: BarthelGernandScript, ClaudeMaricScript, DASheogorathRatScript

OnReset

Этот блок исполняется один раз, когда ячейка с объектом возвращается в исходное состояние (через 3 после последнего посещения игрока). Обычно используется для ловушек и анимированных объектов, чтобы вернуть их в исходное состояние.

В игре Обливион блок-тип OnReset используется 110 раз.
Примеры скриптов: ARCAVEINTRAPSCRIPT01, ARChainPlatform01SCRIPT

OnSell

Синтаксис:
CODE
begin OnSell SellerRefID (не обязательно)

Пример:
CODE
begin OnSell player

Этот блок исполняется один раз, когда заскриптованный объект продается. Если вы указываете копию продавца (SellerRefID), блок будет исполняется, только если указанный объект продает объект; иначе, он выполняется, когда любой NPC продает объект.

В игре Обливион блок-тип OnCell используется 1 раз.
Примеры скриптов: MS21statuescript

OnStartCombat

Синтаксис:
CODE
begin OnStartCombat TargetActorRefID (не обязательно)

Пример:
CODE
begin OnStartCombat player

Этот блок выполняется один раз, когда заскриптованный актер начинает бой с указанной целью (TargetActorRefID). Если цель не указана, блок исполняется, когда актер начинает бой.

В игре Обливион блок-тип OnStartCombat используется 13 раз.
Примеры скриптов: BaurusScript, BurdSCRIPT, CGEmperorScript

OnTrigger

Синтаксис:
CODE
begin OnTrigger TriggeringRefID (не обязательно)

Пример:
CODE

begin OnTrigger player

Этот блок исполняется один раз, когда что-то сталкивается с заскриптованным объектом. Если вы указываете это что-то (TriggeringRefID), блок исполняется, когда указанная копия сталкивается с объектом; иначе блок исполняется, когда любая копия сталкивается с объектом.

В игре Обливион блок-тип OnTrigger используется 94 раза.
Примеры скриптов: CGTriggerZoneCellScript, CGTrigZone01SCRIPT

OnTriggerActor

Синтаксис:
CODE
begin OnTriggerActor TriggeringRefID (не обязательно)

Пример:
CODE
begin OnTriggerActor player

Этот блок исполняется один раз, когда актер сталкивается с заскриптованным объектом. Если вы указываете это актера (TriggeringRefID), блок исполняется, когда указанная копия (актера) сталкивается с объектом; иначе блок исполняется, когда любой актер сталкивается с объектом.

В игре Обливион блок-тип OnTriggerActor используется 11 раз.
Примеры скриптов: CGTrigZoneACTORSCRIPT, FrostFireGasTrap, KillBox01SCRIPT

OnTriggerMob

Синтаксис:
CODE
begin OnTriggerMob TriggeringRefID (не обязательно)

Пример:
CODE
begin OnTriggerMob player

Этот блок исполняется один раз, когда мобильный объект сталкивается с заскриптованным объектом. Если вы указываете этот объект (TriggeringRefID), блок исполняется, когда указанная копия (мобильного объекта) сталкивается с объектом; иначе блок исполняется, когда любой мобильный объект сталкивается с объектом.
Мобильные объекты включают актеров (NPC и существа), стелы и магические снаряды.

В игре Обливион блок-тип OnTriggerMob не используется.

OnUnequip

Синтаксис:
CODE
begin OnUnequip ActorID (не обязательно)

Пример:
CODE
begin OnUnequip
begin OnUnequip player

Этот блок исполняется один раз, когда заскриптованный объект снимается указанным актером (ActorID). Если параметр не используется, блок исполняется, когда объект снимается любым актером.

В игре Обливион блок-тип OnUnequip используется 11 раз.
Примеры скриптов: ArenaRaimentScript, Dark05BladeScript, MGBloodwormHelmScript01

ScriptEffectFinish

Использование:
CODE
begin ScriptEffectFinish

Особый блок используется только в скриптах для магических эффектов. Этот блок исполняется после того, как заскриптованный эффект закончился.

Заметьте, что ScriptEffectUpdate будет исполняться до него. Если в ScriptEffectUpdate при последнем есть команда return, эта часть скрипта не будет исполняться.

В игре Обливион блок-тип ScriptEffectFinish используется 32 раза.
Примеры скриптов: AnvilMGPetImpScript, AtronachFlameScript, AtronachFrostHealSCRIPT

ScriptEffectStart

Использование:
CODE
begin ScriptEffectStart

Особый блок используется только в скриптах для магических эффектов. Этот блок выполняется сразу после того, как заскриптованный эффект наложен.
Примечание: Любое заклинание со скриптовым эффектом исполняется на актере, на которого оно было наложено. Это означает, что копией по умолчанию является актер, на которого наложено заклинание, а не игрок (хотя игрок тоже может быть этим актером). Это также значит, что если блок выполняется, значит заклинание только что наложили на актера и ScriptEffectStart может считаться блоком OnHitWith <имя заклинания>.
Тот факт, что заклинания со скриптовыми эффектами запускаются на объектах, на которые они были наложены, дает программисту возможность запускать скрипты на любом актере на определенный период времени, и может быть использовано для получения копии актера.
В игре Обливион блок-тип ScriptEffectStart используется 42 раза.
Примеры скриптов: AnvilMGPetImpScript, AtronachFlameScript, AtronachFrostHealSCRIPT

ScriptEffectUpdate

Использование:
CODE
begin ScriptEffectUpdate

Особый блок, используется только в скриптах для магических эффектов. Этот блок запускается, как только эффект наложен, и исполняется до тех пор, пока время действия эффекта не закончится.

В игре Обливион блок-тип ScriptEffectUpdate используется 13 раз.
Примеры скриптов: DAHermaeusSoulsSpell, DummyMagicEffectSCRIPT, GhostEffectScript

4.3.6 Команда "." (точка - UseReference)
Синтаксис:
CODE
ObjectID.ObjectFunction [Arguments ...]; IDобъекта.Функция [Аргументы…]

Команда "." устанавливает предыдущий ObjectID (ID объекта) как текущую копию только для следующей функции. ObjectID может быть и копией, и переменной типа reference.

Примечания:
Как бы странно это ни показалось по сравнению с другими командами, но в скомпилированном скрипте это настоящая команда, которая использует код 0x001C.
Выражения (например, QuestID.QuestVariable (IDквеста.КвестоваяПеременная)) не используют эту команду. Внутри игры подобные выражения являются просто сложением (ObjectID)(Variablenumber). Например, строка "MageConvSystem.lecturevar" из скрипта ArcaneUScholarScript в шестнадцатеричном коде будет выглядеть как "72 04 00 73 0F 00" ("MageConvSystem" - четвертая копия, используемая в скрипте, а "lecturevar" объявлена пятнадцатой переменной).
Content - Disposition: form-data; name="smiles_on"

4.3.7 Команда Set (установить)
Присваивает локальной или глобальной переменной указанное значение. Это значение может быть числом или результатом вычисления выражения.

Арифметические операторы:

Оператор | Описание
+ Сложение
– Вычитание
* Умножение
/ Деление
% Модуль (выполняет целочисленное деление и возвращает остаток)

Примечания:
• Оператор модуля "%" вычисляется после умножения / деления, но перед сложением / вычитанием:
4 * 3 % 2 = 0
4 * (3 % 2) = 4
1 + 2 % 3 = 3
(1 + 2) % 3 = 0

• Минус сразу перед числом или переменной считается знаком "отрицания". Если вам необходимо выполнить вычитание, нужно оставить как минимум один пробел до и после минуса. Это единственный случай, когда вам нужен пробел между арифметическими операторами. (Например: "a-b" не будет компилироваться; нужно написать "a - b")
• Когда при делении вы используете только числа, необходимо указать хотя бы один знак после запятой, чтобы показать, что вы хотите использовать деление с плавающей точкой, без него остаток будет отброшен после окончания деления:

CODE
float a
set a to 9/5 ; установит "a" в 1.000
set a to 9.0/5; установит "a" в 1.800

• Если вы хотите сохранить в целочисленной переменной верно округленный результат деления, убедитесь, что в вычислениях используется плавающая точка (и десятичная часть не отбрасывается) и добавьте 0.5:

CODE
>short a
set a to 9/5 ; установит "a" в 1
set a to 9/5 + 0.5; установит "a" в 1
set a to 9.0/5 ; установит "a" в 1
set a to 9.0/5 + 0.5; установит "a" в 2
set a to 7.0/5 + 0.5; установит "a" в 1

CODE
short a
set a to 9
set a to a/5.0 + 0.5; установит "a" в 2

CODE
short a
float b
set b to 9
set a to b/5 + 0.5; установит "a" в 2

• Вы можете использовать сравнение в качестве "значения". Переменная тогда станет равной 1 или 0 в зависимости от того, истинно ли сравнение или нет.

CODE
set goodluck to player.Getav luck > 60

делает то же самое, что и

CODE
if player.Getav luck > 60
set goodluck to 1
else
set goodluck to 0
endif

Вы можете использовать арифметические операции с результатом сравнения, но для этого нужно заключить сравнение в скобки.
Выражение

CODE
set luckbonus to 50 * (player.Getav luck > 60)

делает то же самое, что и

CODE
if player.Getav luck > 60
set luckbonus to 50
else
set luckbonus to 0
endif

Другие примеры:

CODE
set a to 2
set b to a*a
set c to (b - a)*b - a
set d to ((3* -b+a) - c)/ -2
message "a=%.0f, b=%.0f, c=%.0f, d=%.0f" a b c d ; ("a=2, b=4, c=6, d=8")
set stage to getstage quest1 + 10
set weapondrawn to player.isweaponout

4.3.8 Команда Return (возврат)
Синтаксис:
CODE
Return

Return используется для того, чтобы остановить выполнение скрипта в текущем фрейме. Он не просто завершает ТЕКУЩИЙ блок, он завершает весь скрипт до конца фрейма. Скрипт будет запущен снова в следующем фрейме. Это может быть полезно внутри команд if , когда вы можете блокировать команды, следующие после return.

4.3.9 Конструкция из команд If, ElseIf, Else, EndIf.
Команда If (если) позволяет вам исполнять (или не исполнять) группу скриптовых команд в зависимости от того, истинно ли сравнение, которые вы указываете. Команда If в скриптовом языке Обливиона является очень мощной и сравнима с аналогичной командой из “настоящих” языков программирования.

Команда ElseIf (иначе если) используется для увеличения количества проверок условий в одной конструкции If. Если условие не истинно, тогда проверяется условие следующей за ней команды ElseIf. Количество команд ElseIf может быть достаточно большим. Но эта команда не является обязательной.

Команда Else (иначе) используется в конструкции IF в тех случаях, когда ни одна из проверок, объявленных выше, не является истинной. В этом случае будет выполнена группа команд, следующая после нее до команды endif. Команда else, как и elseif, не является обязательной.

Команда Endif обязательна и всегда завершает блок If. То есть, каждому If обязательно соответствует Endif. Обычно все команды конструкции располагают с одинаковым отступом от края страницы для облегчения чтения скрипта.

Синтаксис

Команда if использует следующий синтакис:

CODE
if expressionA [сравнение] expressionB
; проверка "выраженияA [сравнение] выражениеB" пройдена
elseif expressionB [сравнение] expressionC
; проверка "выражениеB [сравнение] выражениеC" пройдена
else
; ни одна из вышеприведенных проверок не пройдена
endif

Операторы сравнения

Команда if может содержать один или несколько операторов сравнения. Вот таблица этих операторов:

CODE
Оператор | Описание
== Равенство (равно)
!= Неравенство (не равно)
> Больше чем
>= Больше или равен
< Меньше чем
<= Меньше или равен

Важно отметить, что в Обливионе нет побитового сравнения.

Объединение сравнений

Сравнения могут быть соединены друг с другом при использовании следующих логических операторов:
CODE
Оператор Описание Пример Описание примера
&& логическое И (and) if x == 1 && y == 1 ; истина, если и x, и y равны 1.
|| логическое ИЛИ (or) if x == 1 || y == 1 ; истина, если ни x, ни y не равны 0.

Заметьте, что " || " рассматривается раньше "&&", также как " * " рассматривается раньше " + " в обычной алгебре.

Если вы хотите, чтобы “&&” рассматривался первым, необходимо использовать скобки. Например:
CODE
if myVar1 == 1 && myVar2 == 1 || myVar2 == 5

Истина, когда MyVar1 = 1 И myVar2 равен 1 ИЛИ 5.

CODE
if (myVar1 == 1 && myVar2 == 1) || myVar2 == 5

Истина, когда или myVar2 равен 5 ИЛИ как myVar1, так И myVar2 равны 1.

Сравнения и выражения

Операторы сравнения работают с любыми выражениями, которые можно представить в числах. Предполагая, что "a = 17", "b = 20" и "c = a - b", все следующие выражения работают, как ожидалось. Скобки необходимы только по математическим причинам:

CODE
IF c == -3 && b == 20
IF c == -3 && b == 20 && a == 17
IF c - 1 == -4 && b == 20 && a == 17
IF a - 20 == 17 - b
IF a - 20 == 17 - b && c + 3 == 0
IF a + 3 == b
IF a - b == c
IF a * 4 - b * 4 == c * 4
IF a * (5 + c) - 14 == b
IF 2 * (a * (5 + c) - 14) == b - - b

Если переменная или результат функции содержат 1 или 0, или вам интересно, равен ли результат 0, вам не нужно проверять "== 1" или "!=0"
Команды
CODE
IF Done
IF Getisid MyNPC
IF Getitemcount Lockpick
IF IsActor && Flag

делают то же самое, что и

CODE
IF Done != 0
IF Getisid MyNPC != 0
IF Getitemcount Lockpick != 0
IF IsActor != 0 && Flag != 0

Примечания.
• Не забывайте, что компилятор не распознает команды, написанные в одну строку. Хотя выражение

CODE
if condition == 1 set varname to 1

else set varname to 2

работает в других языках программирования, в скриптах Обливиона эти команды нужно разделить:

CODE
if condition == 1
set varname to 1
else
set varname to 2
endif

(Кроме того, так более читаемо...)
• После условия или команды "else" остаток строки игнорирует, и никаких сообщений об ошибке не появится, так что (поскольку строка кажется правильной) вы можете потратить много времени, пытаясь обнаружить источник проблемы.

4.4 Переменные (Variables)

4.4.1 Классификация переменных

Ссылки и переменные

Функции оказывают действие на “вызывающий объект” (тот, на котором работает скрипт) по умолчанию, но вы можете вызвать их и на другом объекте, используя ссылку на объект. Когда вы хотите, чтобы скрипт влиял, например, на игрока, используйте ссылку на игрока (“Player”) и “.”, чтобы перенаправить вызов функции (function-call) на игрока с вызывающего объекта:
CODE
player.additem gold_001 100

Для большей гибкости вместо числовых значений вы можете использовать переменные, если они требуются для какой-нибудь функции:
CODE
short addgold
set addgold to 100
player.additem gold_001 addgold

Вы можете использовать локальные переменные, объявленные в этом же скрипте, или же использовать для своих целей глобальные переменные.
Из функции нельзя напрямую обратиться к переменной в другом скрипте. Но такой возможностью обладают команды If и Set.
Например, эта строка скрипта работать не будет:
CODE
player.additem gold_001 otherobject.addgold

Следует написать так:
CODE
short addgold; объявление вспомогательной локальной переменной
set addgold to otherobject.addgold; присвоить ей значение внешней локальной переменной
player.additem gold_001 addgold; использование полученных внешних данные

Переменные можно классифицировать по назначению и по типам.
Переменные по назначению бывают:
• Глобальные
• Специальные
• Локальные
Переменные по типам:
• Short - Короткая целочисленная.
• Long - Длинная целочисленная.
• Float - Вещественная переменная.
• Ref - Переменная типа reference (ссылка на копию объекта).

4.4.2 - Глобальные переменные (Globals)
Глобальные переменные может использовать любой скрипт или условие и они не привязаны ни к какому определенному объекту или квесту.
Вы можете создавать свои собственные глобальные переменные.
Глобальные переменные объявляются в главном меню Gameplay –► Globals.

Описание свойств диалогового окна Globals:
EditorID: Имя переменной. Зарезервированные символы («_», «+», «–», и т.д.) и пробелы использовать нельзя.

Variable Type: Тип переменной Short (короткий целочисленный) и Long (длинный целочисленный) по сути одно и то же. Оба типа являются целочисленными. Float – это вещественная переменная. Внутри игры все переменные хранятся в виде 32-битных переменных с плавающей точкой. Техническая реализация такого формата переменных приводит к неточностям с очень большими и очень малыми значениями. (например, все числа от 2000000000 до 2000000064 хранятся как 2000000000).
Value: Значение переменной по умолчанию. Переменная будет иметь такое же значение, как и сразу после установки плагина. После этого значение переменной будет храниться в файле сохраненной игры.

4.4.3 - Перечень глобальных переменных Обливиона
Здесь приведены все глобальные переменные, которые используются в игре.

Глобальные переменные, обрабатываемые Oblivion.exe:
GameDay - Возвращает внутриигровой день.
GameDaysPassed - Возвращает количество дней, прошедших с начала игры.
GameHour - Возвращается внутриигровой час
GameMonth - Возвращает внутри игровой месяц
GameYear - Возвращает внутриигровой год
Timescale - Отношение игровой минуты к минуте реального времени (по умолч. – 30)

Глобальные переменные, обрабатываемые в скриптах и диалогах:
CrimeForceJail
EmfridBittneld
Fame
FameAkatosh
FameArkay
FameDibella
FameJulianos
FameKynareth
FameMara
FameStendarr
FameTiberSeptim
HasGrace
HighwaymanGotMoney
KvatchDestroyed
MQ05RavenPlace
MQEndDayWeek
MS05InProgress
MS22DealGold
MS22RewardGold
MS38LastContact
OblivionCrisis
PCVampire:
Сохраняет стадию вампиризма игрока. Возможные значения:
0: Не вампир (по умолчанию)
1-4: Стадии вампиризма.
-1: Игрок вылечился от вампиризма.
Используется в скриптах игры:
- VampireScript
- MS40PotionEffect
RentAnvilCountsArms
RentAnvilFlowingBowl
RentBorderWatchInn
RentBravilLonelySuitor
RentBravilSilverHome
RentBrinaCross
R entBrumaJerallView
RentBrumaOlavs
RentCheydinhalBridgeInn
RentCheydinhalNewlandsLodge
RentChorrolGreyMare
RentChorrolOakandCr osier
RentDrunkenDragon
RentFaregyl
RentGottshawInn
RentHackdirtMoslinsInn
RentICAllSaintsInn
RentICBloatedFloat
RentIC KingandQueen
RentICLuthorBroad

RentICMerchantsInn
RentICTiberSeptimHotel
RentIllOmen
RentImperialBridge
RentLeyawiinFiveCl aws
RentLeyawiinThreeSisters
RentNewlandsLodge
RentRoxeyInn
RentSkingradTwoSistersLodge
RentSkingradWestWeladInn
RentWawne tInn
Sleep
TGPayoffCrimeGold
TGPriceAttack
TGPricePerKill
TGPriceSteal
UmbacannoCount

4.4.4 - Специальные переменные (Special variables)
Специальные переменные могут относиться как к глобальным, так и к локальным.
Некоторые глобальные переменные заданы с самого начала и их значения обновляет сама игра.
В таблице приведены такие виды глобальных переменных:
CODE
Тип Наименование Описание
float GameYear Текущий год
float GameMonth Текущий месяц
float GameDay Текущий день
float GameHour Текущий час (0-24 часов)
float TimeScale Сколько минут игры проходит за одну минуту реального времени
float GameDaysPassed Кол-во дней, прошедших с начала игры

CODE
Значение GameMonth | Название месяца | Описание
0 Утренней Звезды Кол-во дней как в январе (31)
1 Восхода Кол-во дней как в феврале (28)
2 Первоцвета Кол-во дней как в марте (31)
3 Дождя Кол-во дней как в апреле (30)
4 Сева Кол-во дней как в мае (31)
5 Середины Года Кол-во дней как в июне (30)
6 Солнцеворота Кол-во дней как в июле (31)
7 Урожая Кол-во дней как в августе (31)
8 Огня Кол-во дней как в сентябре (30)
9 Мороза Кол-во дней как в октябре (31)
10 Заката Кол-во дней как в ноябре (30)
11 Вечерней Звезды Кол-во дней как в декабре (31)

Есть также локальные переменные с особыми функциями:

CODE
Тип скрипта | Тип перем | Имя | Описание
Квест float QuestDelayTime Определяет, как долго (в реальных секундах) игра будет ждать между двумя исполнениями квестового скрипта. По умолчанию, когда fQuestDelayTime установлена в 0, это значение составляет 5 сек. Если задать очень малое значение, например, 0.01, то квестовый скрипт будет исполняться в каждом фрейме.

Объект (ловушка) float fTrapDamage Количество повреждений, которые наносятся каждый раз, когда ловушка воздействует на актера.

Объект (ловушка) float fTrapPushBack Количество ударной силы, которая применяется к актеру. Должна быть в диапазоне от 0 до 1000.

Объект (ловушка) float TrapMinVelocity Минимальная скорость, с которой ловушка должна двигаться относительно игрока, чтобы нанести повреждения. Значение в BSUnits (128 = 6 футов).

Объект (ловушка) float bTrapContinuous 0 = Наносить повреждения только при первом контакте с ловушкой. 1 = Постоянно наносить повреждения, пока контакт с ловушкой сохраняется.

4.4.5 Типы переменных

Есть несколько типов переменных. Хотя вы можете объявить переменную, как short или long, результат будет одинаковым, если вы не выходите из диапазона.

short - Короткая целочисленная
Диапазон: от -32768 до 32767
Пример: short varNameShort

long - Длинная целочисленная
Диапазон: от -2 147 483 648 до 2 147 483 647
Пример: long varNameLong

float - Вещественная переменная
Диапазон: от -3.402823×10^38 до -1.175494×10^−38,
0 и от 1.175494×10^−38 до 3.402823×10^38
(точность до 7 знаков)
Пример: float varNameFloat

ref - Ссылка на копию объекта
Диапазон: 32-битный код FormID
Пример: ref varNameRef

4.4.6 Целочисленные короткие переменные (Variable types: shortint)
Короткие целочисленные переменные (short) – это целочисленный формат переменных, использующий 16 бит. Самый старший бит – это знак, поэтому диапазон значений переменной от -2^15 (-32768) до (2^15)-1 (32767).

4.4.7 Целочисленные длинные переменные (Variable types: longint)
Длинные целочисленные переменные (long) – это целочисленный формат переменных, использующий 32 бита.
Самый старший бит – это знак, поэтому диапазон значений переменной от -2^31 (-2147483648) до (2^31)-1 (2147483647).
Когда вы используете long-переменную в скрипте, нет никаких ограничений, но если это глобальная переменная, она хранится как float и могут возникнуть проблемы с точностью при крайних значениях.

4.4.8 Вещественные переменные
(Variable types: floating point)

Информация из WIKI:
Вещественные переменные (float) - это формат чисел, в котором можно хранить и очень большие, и очень малые значения.
Диапазон составляет от 1.18E-38 до 3.40E38.
Однако этот формат не всегда является лучшим выбором, потому что техническая реализация этого формата приводит к некоторым неточностям (например, все числа от 2 000 000 000 до 2 000 000 064 хранятся как 2 000 000 000).

Почему это происходит?
В реальной жизни вещественные числа имеют основание 10. Компьютер использует основание 2, что делает все несколько сложнее, но, по сути, остается прежним, поэтому для объяснений мы будем использовать основание 10.
В школе нас учили как записывать большие числа, используя экспоненту, чтобы экономить место и улучшать читаемость: "1 000 000 000" - то же самое, что и " 10^9 ", а "1 234 000 000" – это " 1,234 * 10^9 " или " 1,234E9 ". Также и с малыми числами: " 0,000001234 " - это "1,234E-6"
Когда длина первого числа (мантиссы) ограничена, очень большие и очень малые числа обрезаются. Обливион использует около 7 знаков для мантиссы:
1.0000000E9 = 1 000 000 000
1.0000001E9 = 1 000 000 100
Таким образом, нельзя отображать числа между 1 000 000 000 и 1 000 000 100 потому, что они требуют 9 знаков, а сохраняется только первые семь.

Прим. Garin:
На самом деле приведенные примеры не совсем точны. Смотреть надо двоичный код в районе 7-ми десятичных разрядов - это число должно быть кратно двоичному коду. Нужно дополнительно поэкспериментировать, чтобы определить точно это число. Ориентировочно 24 двоичных разряда.
Например, числу 10 000 000 соответствует двоичный 24-хразрядный код 1001 1000 1000 1011 0100 0000.
Двоичный 24-хразрядный код 1000 0000 0000 0000 0000 0000 соответствует десятичному 8388608 или 8.388608E6.
Максимальное число для 24 разрядов – это 16 777 215. Вот тут и надо ловить. Все, что больше, в младших разрядах мантиссы обрезается.
То же самое было в TES3. Там длинные 32-разрядные двоичные числа в глобальных переменных "обрезались" до 24-х разрядов, а вот в локальных почему-то работали.
Давайте рассуждать логически. Допустим, вещественное число занимает 4 байта или 32 двоичных разряда. Из них 3 байта отведены на мантиссу, а 1 байт - на степень. 3 байта мантиссы - это 24 двоичных разряда. 1 байт на степень - это 255, но в игре - только (- 38...+38).
Не стоит забывать также, что старший разряд отдан для знака.
4.4.9 Переменные для хранения копий (Reference variable)

Как и другие переменные, переменные для хранения копий (ref) должны быть объявлены до их использования:
CODE
ref refVarName

Их можно использовать во всех функциях, которые работают с копиями объекта. Неинициализированные ref переменные указывают на сам объект, в скрипте которого они были объявлены; в остальных случаях они ссылаются на указанный объект.
Ref-переменная, объект которой не находится в памяти (например, статический объект, когда игрок вышел из ячейки, где он находится) считается неинициализированным.
Вы можете присвоить значение ref-переменной, используя команду set. Обычно для этого вы будете использовать функцию, возвращающую formID. Например:
CODE
set myRef to GetContainer

Можно задавать значения переменных, объявленных на другом объекте:
CODE
set BobRef.myRef to GetSelf

Проверка того, что переменная не инициализирована:
CODE
if myReferenceVariable == 0

Чтобы проверить, инициализирована ли переменная:
CODE
if myReferenceVariable != 0

Чтобы проверить, указывает ли переменная на другую копию (например, на игрока):
CODE
if myReferenceVariable == player

или другую копию:
CODE
if myReferenceVariable != player

Пример:
CODE
scn TheMastersSword ; кто оденет, получит +50 к навыку "Мечи"

ref UserRef

begin OnEquip
set UserRef to GetContainer
UserRef.Modactorvalue blade 50
end

begin OnUnEquip
set UserRef to GetContainer
UserRef.Modactorvalue blade -50
end

Помните, что ссылки на непостоянные предметы недоступны, когда объект не находится в памяти (например, в другой ячейке). То же самое относится к предметам, находящимся в контейнере. Попытка доступа к недоступным переменным может не иметь никакого эффекта, а может привести к вылету. В CS есть механизмы, не позволяющие прямо запрашивать доступ к подобным объектам, но вы можете обойти их с помощью ref-переменных.
Делайте это на свой страх и риск...

5. Ваш второй скрипт
Мы включили это руководство из WIKI полностью. Оно содержит уже известные вам основы скриптинга, но мы сочли, что это даже хорошо, поскольку «повторение - мать учения».

5.1 Введение
Учебник WIKI "Мой первый скрипт" хорош для первого ознакомления со скриптами, но он не раскрывает в полном объеме всех возможностей скриптового языка Oblivion. Это прекрасное введение для тех, кто никогда раньше не имел дела с программированием или скриптами, но модмейкерам будет полезен и более углубленный учебник для дальнейшего изучения этого замечательного ресурса.

Данный учебник в основном был адаптирован из "Руководства GhanBuriGhan’a по скриптам Morrowind для чайников" (GhanBuriGhan's excellent Morrowind Scripting for Dummies); так что авторские права принадлежат GhanBuriGhan за проделанную им фантастическую работу над оригиналом.

Этот учебник должен послужить более полным введением в скрипты для Oblivion, чем "Мой первый скрипт", и рассчитан на то, что читатель уже знаком с этим руководством. Если вы не поняли его основные принципы, то данный учебник может оказаться для вас слишком сложным. Но если с "Моим первым скриптом" у вас все в порядке, тогда начинаем!

5.2 Содержание

5.3 Информация о написании скриптов в Oblivion
Что такое скрипт?
Скрипты – это базовые кусочки кода, написанные на специальном скриптовом языке (далее - скрипты TES). Эти маленькие "программы" будут работать во время игры и могут выполнять определенные вещи, действительно много вещей: события на триггерах, контроль времени и места, исчезновение предметов и существ, появление или движение, выдача сообщений игроку, изменение параметров и даже изменение погоды – возможности просто невероятны.

Скриптовый язык TES уникален, но он не может быть использован вне TES Construction Set. Как скриптовый язык он имеет некоторые ограничения, в отличие от "настоящих" языков программирования, например C++:
Область применения скриптов TES ограничена; не надо ждать, что можно запрограммировать что-то, так или иначе не включенное в игру. Это совсем не означает, что вы не сможете достичь новых и необычных результатов! Но вы не сможете использовать скрипты TES, например, для программирования текстового процессора.
Скрипты TES не похожи на SDK (комплект разработки программного обеспечения – сокр. с англ. software development kit), который действительно позволяет работать и изменять исходные коды игры. Вот почему вы не сможете использовать скрипты TES, например, для добавления новых погодных эффектов. Они прописаны в другом месте и нужно обладать дополнительными знаниями, чтобы сделать это.
Данный язык интерпретируемый, а не компилируемый – коду для работы нужна отдельная программа (в данном случае Oblivion.exe), в отличие от компилированного кода, который может работать сам по себе, как .exe-приложение.
TES скрипты не зависят от регистра. Это означает, что команда "player.getpos z" будет работать точно так же как и "Player.GetPos z" или "PlAyEr.GeTpOs Z", или любой другой возможный вариант. Многие (как и автор) используют второй вариант, т.к. он наиболее понятен; множество других людей используют первый вариант, т.к. проще писать, используя один регистр. Единственно регистр имеет значение, когда вы хотите напечатать сообщение, выводимое на экран: но даже там все зависит от языка и от цели сообщения, регистр может быть любой.

Что могут скрипты?
Скрипты в Oblivion – способ, с помощью которого игра динамически реагирует на то, что делает игрок в игровом мире. Вы можете использовать скрипты для создания комплексных квестов. Вы можете использовать скрипты для создания особых предметов, которые будут выполнять действия, невозможные для обычного зачарования. Вы можете использовать скрипты для создания ловушек. Вы можете использовать скрипты для изменения поведения NPC или существ. Помните создание персонажа в Oblivion? Оно контролируется множеством скриптов. Выполнение квестов? Они также управляются скриптами. Так что если коротко ответить на этот вопрос: много.

Чего не могут скрипты

Скриптовый язык TES ограничен в своих возможностях – в нем большое количество функций, которые вы можете использовать, но иногда возможные варианты использования будут не все, которых бы вам хотелись. В большинстве случаев опытные скриптологи смогут найти специфику для очевидных ограничений, но не стоит ожидать чуда. Многие вещи запрограммированы и скрипты на них влияния не оказывают, а если и оказывают, то лишь косвенно.

5.4 Учебник скриптов: до написания кода.
Если вы новичок в скриптах и в программировании разбираетесь лишь в общих чертах, даже если уже ознакомились с учебником "Мой первый скрипт", углубленное скриптонаписание с использованием скриптов TES может немного напугать. Вот почему здесь представлен развернутый учебник, который постепенно подведет вас к созданию более сложного скрипта. По мере изучения вы получите объяснения основных элементов скриптового языка. Вам встретятся разнообразные пояснения, но ключевые инструкции и важная информация будут выделены жирным шрифтом.

Начинаем!
Начинаем мы с открытия редактора скриптов: Запустите TES Construction Set, откройте Oblivion.esm, затем выберите Edit Scripts в меню Gameplay, чтобы открыть окно редактирования скриптов.

Окно редактирования скриптов
Открыть окно редактора скриптов можно несколькими способами: выбрав Gameplay –> Edit Scripts; нажатием на кнопку редактора скриптов (карандаш) в правом верхнем углу панели инструментов; или в меню Object или NPC, активировав кнопку […], расположенную за редактируемым полем с названием скрипта.

Давайте посмотрим на кнопки, расположенные на панели инструментов, слева направо:
Открыть - позволяет выбрать скрипт для редактирования.
Сохранить - проверяет текущий скрипт и компилирует его или выдает сообщения об ошибках. Примечание: в это время реального сохранения плагина на диск не происходит. При написании больших скриптов после сохранения скрипта вы должны периодически использовать команду "сохранить плагин" в главном окне TES CS, на случай "падения" конструктора. Также если при редактировании скрипта вы вдруг нажмете "сохранить плагин", изменения в скрипте при этом сохранены НЕ будут. Сначала вы должны сохранить вручную сам скрипт. Также, если просто закрыть скриптовое окно, это не значит, что скрипт сохранится. Вы должны позаботиться о его сохранении сами.
Стрелки Вперед и Назад открывают следующий или предыдущий скрипт соответственно (в алфавитном порядке). Если вы присвоите своим скриптам общее обозначение в начале их названий, то это облегчит переход между скриптами и их поиск в вашем проекте. Например, мой псевдоним - Grundulum, и он стоит в начале каждого моего скрипта "GR_ShortReferencetoProject_", а последующие две или три буквы образуют аббревиатуру для конкретного мода; это заставит все скрипты, над которыми вы работаете, расположиться аккуратно и по порядку.
Скомпилировать все – перекомпилирует все скрипты (для чего это нужно? Ни я, ни GhanBuriGhan не знаем). Это также добавит каждый скрипт из Oblivion в ваш мод - таким образом вы получите двухмегабайтный esp-файл, конфликтующий практически со всем, с чем только можно. Используйте это кнопку осторожно.
И, наконец, кнопка Удалить – удаляет скрипт, а последняя кнопка "Стрелка вниз" закрывает скриптовое окно.
В правом верхнем углу панели инструментов можно найти ниспадающий список под названием Тип Скрипта. Данный блок позволяет выбрать один из трех типов скриптов, под который подпадает ваш скрипт: Объект, Квест или Магический Эффект. Подробнее об этом позже, но в рамках введения - Объектные скрипты – это скрипты, привязанные к объектам игрового мира (также, как и к предметам или NPC), Квестовые скрипты – скрипты, контролирующие выполнение квестов (как генерация персонажа), и Магические скрипты – скрипты, контролирующие все особые магические эффекты (в особенности, скриптовые эффекты).

Что мы хотим?
Перед тем, как мы приступим к написанию нашего учебного скрипта, мы сначала должны решить, что мы хотим, чтобы он делал!

Для этого учебника мы собираемся создать:
Шкаф с Загадкой: шкаф загадывает загадку и открывается только в случае правильного ответа. Если игрок отвечает неправильно – срабатывает ловушка и ранит игрока, а шкаф остается закрытым.
Это довольно сложное задание, но мы будем выполнять его шаг за шагом и в конце станет очевидно, что не все так страшно.

Написание скрипта
После открытия скриптового окна, нажмите Script -> New.... Вы должны заметить, что Тип Скрипта – "Объект", что нам как раз и нужно. Нажмите в главном окне, которое из серого теперь стало белым. Здесь вы будете писать скрипт.

5.5 Учебник скриптов: первые строки.
Называем скрипт
Сначала мы должны назвать скрипт. Каждый скрипт должен начинаться с его названия. В окне редактора, пожалуйста, напечатайте

CODE
ScriptName RiddleChestScript

Отметьте, что ни пробелов, ни подчеркиваний в названии нет. Название скрипта должно состоять из одного слова, так что использовать пробелы нельзя. Также Oblivion игнорирует подчеркивание в алфавитном списке скриптов; вы можете

использовать подчеркивание, чтобы сделать название более понятным для себя, но вы не увидите его, когда позже откроете скрипт. (Дополнительная информация: Oblivion допускает так называемый эффект наложения в скриптах TES. Это когда вы заменяете полное имя команды его сокращенной версией. Для примера: "ScriptName" становится "scn", или "ForceActorValue" становится "ForceAV". Мы не будем использовать эффект наложения в данном учебнике, но не удивляйтесь, если увидите подобное в других скриптах.) Помните, что скрипты TES не чувствительны к регистру; вы могли бы написать "scriptname" или "ScriptnamE", как угодно.

Попробуйте сохранить свой скрипт, используя кнопку «Сохранить» на панели задач. Ничего очевидного не случится, но если вы нажмете на кнопку «Открыть», то можете увидеть название вашего скрипта в перечне скриптов Oblivion. Нажмите на «X» в верхнем правом углу окна выбора скриптов, чтобы закрыть его и вернуться в окно редактирования. RiddleChestScript сейчас самый короткий скрипт в Oblivion, но он ничего не делает. Давайте это исправим.

Команды "Begin" и "End"
Команды "Begin" и "End" определяют начало и конец блоков скрипта – отдельных участков кода, которые будут выполняться при определенных условиях. Например, "Begin GameMode" определят начало выдачи текстового блока, который будет запускаться в каждом фрейме до тех пор, пока игрок не откроет меню. Для наших целей нам нужен тип-блок "OnActivate": он запустится только тогда, когда наш шкаф будет активирован (утомительно отвечать на загадку в каждом фрейме или всякий раз, как NPC умрет, обе этих вариации возможны с использованием других вариантов типов блоков Begin!).
Нажмите два раза Enter, чтобы переместить курсор вниз, и отредактируйте свой скрипт, чтобы он выглядел вот так:
CODE
ScriptName RiddleChestScript

Begin OnActivate
; Здесь мы напечатаем, что случится, когда шкаф будет открыт
End

Опять нажмите "Сохранить". Скрипт все еще ничего не делает, но, по крайней мере, теперь мы определили условия, при которых он будет работать. Также здесь нужно отметить пару вещей. Первое - мы закрыли текущий блок перед тем, как начали новый – это не так важно, когда в скрипте один блок, но как только скрипт станет более сложным, нам понадобятся составные блоки. Выражаясь в более технических терминах – блоки Begin/End не вкладываемые.
Вторая вещь, которую надо отметить, это точка с запятой ";" на линии 4. Точка с запятой отмечает остальную часть строки как комментарий. Все напечатанное после точки с запятой на этой строке при компилировании скрипта будет проигнорировано. Комментарии используются, чтобы объяснить код, это поможет вам и другим быстро понять, что происходит в скрипте на определенных строках. Если вы хотите написать более чем одну строку с комментарием, то точку с запятой нужно ставить в начале каждой строки.

Выдача текстового сообщения и получение ответа от игрока.

Теперь нужно, чтобы наш шкаф с ловушкой загадывал игроку загадку. Для этого используется функция "MessageBox", которая должна быть вам знакома: в " Моем первом скрипте" использовалась родственная команда "Message". "MessageBox" позволяет вывести на экран небольшое количество текста и также отобразить несколько вариантов ответов, из которых может выбрать игрок. К сожалению, в Oblivion (как до него и в Morrowind), нет возможности печатать ответ на загадку, так что нам придется предоставить несколько вариантов ответов. Строка кода для этого:
CODE
MessageBox "Безголосый, но плачет; бескрылый, но парит; беззубый, но кусает;
безротый, но бормочет. Что это?", "Летучая мышь", "Старуха", "Ветер", "Привидение"

Первая строка (текст между первой парой кавычек) – текст, действительно отображаемый в окне сообщения; остальной текст, разделенный запятыми, сообщает игре, чтобы она сделала "кнопки" по одному выражению для каждой.

Но как нам сделать, чтобы загадка задавалась только при первой попытке открыть шкаф, а не постоянно? Теперь мы переходим к центральной теме: использование одноразовых условий и переменных состояния. Самая распространенная из проблем, с которой сталкиваются начинающие в написании скриптов для Oblivion, берет свои корни в непонимании того, как на самом деле выполняются и, соответственно, должны быть оформлены скрипты. Так что обратим внимание на эту важную деталь.

Как выполняются объектные скрипты

Каждый скрипт, привязанный к объекту или NPC (объектный скрипт), выполняется в каждом фрейме отображения игры на экране, пока активна ячейка с объектом (во внутренних ячейках скрипты активны только в ячейке, в которой находится игрок; снаружи – в ячейке, в которой находится игрок и во всех активных смежных ячейках). Так что каждый отдельный скрипт (а не его отдельная строчка) выполняется 10-60 раз в секунду или столько раз, насколько быстро на вашем компьютере идет игра! Легче всего представить каждый локальный скрипт завернутым в большую циклическую спираль:

до тех пор пока (объект в активной ячейке)
[код скрипта]
завершение

Вот почему представленный ниже скрипт будет плеваться непрерывным потоком сообщений (если приложен к объекту или NPC в той же ячейке, где и игрок). Проверьте сами, если хотите:
CODE
ScriptName HorribleMessageScript

Begin GameMode
MessageBox “Тысячи бесполезных сообщений!”
End

Этот пример сравнительно безобидный, но представьте себе что случится, если бы вы использовали линию кода, которая добавляет предмет в инвентарь игрока или устанавливает возле него монстра, и т.п.!

Вот почему конструкции с использованием "Do Once" так существенны и часто используются при написании скриптов для Oblivion. Так что давайте продолжим написание нашего учебного скрипта - нам нужно применить переменную, чтобы наше сообщение выдавалось только один раз. Измените свой скрипт как указано ниже:
CODE
ScriptName RiddleChestScript

Short controlvar

Begin OnActivate
If (controlvar == 0)
MessageBox "Безголосый, но плачет; бескрылый, но парит; беззубый, но кусает;
безротый, но бормочет. Что это?", "Летучая мышь", "Старуха", "Ветер", "Привидение"
Set controlvar to 1
EndIf
End

и опять сохраните скрипт. Заметьте, что команда "MessageBox" должна быть написана на одной строке! Не прерывайте и не переносите ее!

И еще, напоследок нужно отметить несколько вещей. Команда "If" используется для проверки условия – всякий раз, когда выражение в круглых скобках будет истинным, будут выполняться последующие строки кода, вплоть до команды "EndIf". "==" – проверяет, эквивалентно ли левое выражение (в нашем случае это переменная "controlvar") выражению с правой стороны (в нашем случае "0"). Если вы забудете поставить "EndIf" после команды "If", редактор будет выдавать сообщение об ошибке, когда вы попытаетесь сохранить скрипт. "Short controlvar" указывает новую переменную, которую мы назвали "controlvar". Сейчас вам достаточно знать, что данная переменная будет содержать целые (целые положительные или отрицательные числа). Переменная - это "метка-заполнитель", которая может принимать различные значения. Команда "Set" для вас тоже новая, но достаточно простая – она устанавливает значение нашей переменной с 0 на 1 (все переменные начинают с нулевого значения). Это в сочетании с командой "If (controlvar == 0)" обеспечивает одноразовое условие – в следующем фрейме скрипт будет выполняться после того, как переменной будет присвоено значение 1, условие "If" будет неверно и текстовое сообщение больше выдаваться не будет.

5.6 Учебник скриптов: первый тест

Сохранение и подготовка модификации.

Наш скрипт уже способен работать, так что давайте его проверим:

• Сохраните скрипт и закройте окно редактора скриптов.
• Посмотрите в окно объектов и проследуйте по директориям WorldObjects > Container > Clutter пока не найдете объект с Editor ID: CupboardFoodLower.
• Теперь или щелчком с зажатой правой кнопкой мыши выберите Edit, или просто щелкните два раза по объекту, чтобы открыть его свойства.
• Затем в ниспадающем списке скриптов выберите только что сделанный нами RiddleChestScript.
• Нажмите ОК, сохраните мод и выйдите из TES CS.

ПРИМЕЧАНИЕ: ОБРАТИТЕ ВНИМАНИЕ НА ТО, ЧТО МЫ ТОЛЬКО ЧТО СДЕЛАЛИ. РЕДАКТИРОВАНИЕ ОБЪЕКТА БЕЗ ПРИСВОЕНИЯ ЕМУ НОВОГО ID ОПАСНОЕ ЗАНЯТИЕ. НИКОГДА ТАК НЕ ДЕЛАЙТЕ, ЕСЛИ ВЫ НЕ УВЕРЕНЫ В ТОМ, ЧТО ВЫ ДЕЛАЕТЕ!

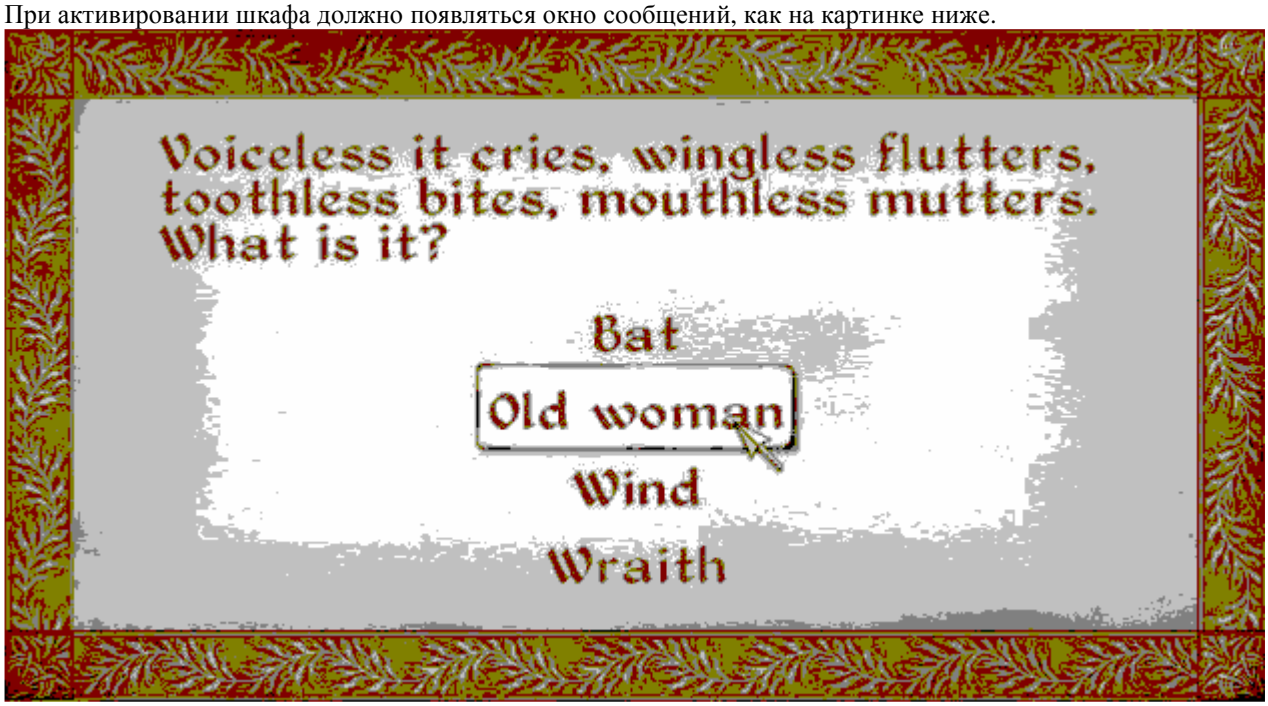
Скрипт в игре
• Теперь используйте "Файлы данных", чтобы активировать мод, запустить игру и загрузить сохранение.
• Когда игра загрузится, активируйте консоль (обычно это кнопка ~ (тильда), слева от 1 на основной клавиатуре) и в консоли напечатайте:
CODE
player.coc "AleswellInn"

и нажмите Enter.

Тестовая локация, шкаф в Алесвельской таверне

• Вы появитесь в Алесвельской таверне, где вас сразу поприветствует группа невидимых людей. Это не имеет отношения к моду, просто данная ячейка использовалась автором для тестирования (она находится сверху в списке интерьеров, поэтому ее очень легко находить). Щелкайте, пока не сможете двигаться, и подойдите к шкафу слева от вас.

При активировании шкафа должно появляться окно сообщений, как на картинке ниже.



Созданное нами окно сообщений

Нажатие на один из вариантов ответа закроет окно, и если попытаться активировать шкаф опять, ничего больше произойти не должно – это хорошо, т.к. значит, что наша проверка состояния работает, как мы и хотели. (Шкаф не открывается потому, что мы еще не включили в скрипт команду Activate, мы сделаем это позже.)

Выключите Oblivion и загрузите свою модификацию в TESCS. Бедным людям из Алесвельской таверны придется дождаться следующего раза, что бы вернуть свою видимость, но такова жизнь NPC в видеоигре.

5.7 Учебник скриптов: выбор игрока, ошибки и исправления.

Выбор игроком варианта ответа

Теперь нужно, что бы скрипт вычислял, какой вариант ответа выбирает игрок и соответственно реагировал на правильный и неправильные ответы. Функция для проверки выбранного ответа "GetButtonPressed". Данная функция устанавливает число в зависимости от того, на какую кнопку щелкнуть мышкой. Устанавливает "0" для первой кнопки ("Летучая мышь" в нашем примере) и 1, 2, 3, и т.п. для следующих кнопок, в порядке, указанном в команде MessageBox. Пока ответ не выбран, функция устанавливает 1, об этом нам тоже следует позаботиться.
Функция "Activate" заставит наш шкаф открыться; фактически, эта функция просто инициирует стандартное действие, которое должно выполняться, когда вы "используете" объект со скриптом: дверь откроется, NPC начнет диалог и т.п.
Следующие изменения в скрипте также демонстрируют, как можно использовать управляющую переменную, чтобы заставить Oblivion обрабатывать функции одну после другой, хотя законченный скрипт обрабатывается каждый фрейм игры: просто увеличьте управляющую переменную и проверьте его в серии операторов If–ElseIf. Это очень безопасный способ скриптонаписания для Oblivion. Это может быть не всегда нужно, но безопасно.

Пожалуйста, измените свой скрипт, как указано ниже:
CODE
ScriptName RiddleChestScript

Short controlvar
Short button

Begin OnActivate
If (controlvar == 0)
MessageBox "Безголосый, но плачет; бескрылый, но парит; беззубый, но кусает;
безротый, но бормочет. Что это?", "Летучая мышь", "Старуха", "Ветер", "Привидение"
Set controlvar to 1
ElseIf (controlvar > 1)
Activate
EndIf
End

Begin GameMode
If (controlvar == 1)
Set button to GetButtonPressed
If (button == -1)
Return
ElseIf (button == 2)
MessageBox "Правильный ответ."
Activate
Set controlvar to 2
Else
MessageBox "Неправильный ответ."
Set controlvar to 0
EndIf
EndIf
End

Обратите внимание на новый блок - "GameMode", который начинается с "If (controlvar == 1)". Мы устанавливаем "controlvar" в 1 как только шкаф будет активирован. Но "OnActivate" только запускает скрипт в том фрейме, в котором объект будет активирован. Поэтому нам необходимо использовать блок "GameMode", чтобы продолжить проверку выбора игрока, и далее проверка будет выполняться в каждом фрейме. Теперь проверим нажатие кнопок. А сделаем мы это с помощью присвоения новой переменной "button" величины, возвращаемой "GetButtonPressed". Скрипт работает, даже когда игра останавливается в ожидании вашего решения. Сначала мы проверим ситуацию, когда ни одна кнопка не была нажата – в этом случае команда "return" сообщает игровому движку остановить выполнение скрипта в текущем фрейме.
Правильный ответ - "Ветер"; он соответствует кнопке номер два – при нажатии этой кнопки нам нужно известить игрока, что он дал правильный ответ, и позволить функции "Activate" открыть шкаф обычным способом. Все значения, соответствующие остальным кнопкам, означают, что игрок выбрал неправильный ответ, так что здесь мы можем использовать команду "Else". В этом случае мы сообщим игроку, что он глупый и шкаф не будет активирован.
Теперь посмотрите на небольшое дополнение сверху скрипта:
CODE
Begin OnActivate
If (controlvar == 0)
MessageBox "Безголосый, но плачет; бескрылый, но парит; беззубый, но кусает;
безротый, но бормочет. Что это?", "Летучая мышь", "Старуха", "Ветер", "Привидение"
Set controlvar to 1
ElseIf (controlvar > 1)
Activate
EndIf
End

Это означает, что если шкаф будет активирован в будущем, он откроется только если "controlvar" будет больше 1. Вспомним последний абзац: если игрок дает неправильный ответ на загадку, "controlvar" получает значение 1, так что он никогда больше не сможет открыть шкаф. Но если ответ правильный, controlvar получает значение 2, и в дальнейшем игрок сможет открывать шкаф сколько угодно. Сохраните и запустите свой плагин, проверьте написанное выше.

Первые ошибки и исправления
Если вы тестировали плагин, то должны были обратить внимание, что все пошло не совсем так, как было запланировано. Выбор неправильного ответа не запирал шкаф навсегда, а если игрок выбирал правильный вариант, то инвентарь и сообщение о правильном ответе отображались одновременно. Сообщение должно быть закрыто перед тем, как можно будет сделать что-то еще, но оно прикрыто инвентарем!

Давайте попробуем следующее (измените соответствующую секцию скрипта, чтобы он стал похожим на этот:
CODE
If (controlvar == 1)
Set button to GetButtonPressed
If (button == -1)
Return
ElseIf (button == 2)
MessageBox "Правильный ответ."
Set controlvar to 2
Else
MessageBox "Неправильный ответ."
Set controlvar to 0
EndIf
ElseIf (controlvar == 2)
Activate
EndIf

Видите, как мы переместили команду "Activate" в секцию, которая проверяет "controlvar == 2"? Это обеспечивает четкую последовательность событий, предотвращая одновременное открытие двух блоков, а как было упомянуто выше, четкая последовательность может быть очень важна при написании скриптов для Oblivion – всегда старайтесь избегать делать слишком много вещей сразу! Сохраните, запустите и проверьте скрипт.

Замечательно, теперь инвентарь открывается как надо, но что это? Мы не можем его закрыть! Посмотрите выше: переменной "controlvar" была присвоена 2, и она и далее остается с таким значением, поскольку мы ее больше не изменяем. Следовательно, игра теперь получает непрерывные команды "Activate" всякий раз, как выполняется скрипт (в каждом фрейме)! Вот почему мы не можем закрыть инвентарь – он мгновенно открывается снова. Так что измените следующую часть скрипта:
CODE
ElseIf (controlvar == 2)
Activate
Set controlvar to 3
EndIf

Заново протестируйте мод: теперь все работает так, как мы хотели. Надеюсь, вас не смутило вышеуказанное отклонение в процесс отладки, но это очень важная вещь – вам постоянно придется продумывать свои скрипты и испытывать различные способы, чтобы добиться успеха.

Что мы пропустили? Ловушку, конечно!

5.8 Учебник скриптов: добавляем ловушку.

Если игрок неправильно ответит на загадку, наш шкаф наложит на него проклятье. Сначала выберите заклинание, которым хотите поразить игрока: щелчок на значок "+" рядом с Magic, еще раз нажмите на "+" рядом со вкладкой Spell, наконец, выберите подпункт "Заклинание". Выбор здесь довольной большой, но для наших целей мы используем Mg05FingerSpell15 spell.

Выбрав это болезненное наказание, нужно применить его к игроку, когда он даст неправильный ответ. Отредактируйте скрипт:
CODE
Else
MessageBox "Неправильный ответ."
Cast Mg05FingerSpell15 Player
Set controlvar to 0
EndIf

Отметьте, что мы использовали функцию "Cast". (Дополнительная информация: для работы функции "Cast" необходим объект вызова. Мы не включили его в наш скрипт, потому что это объектный скрипт, и он будет привязан к игровому объекту. Следовательно, функция будет действовать относительно данного объекта.)

Теперь ваш скрипт должен выглядеть так:

```
CODE
ScriptName RiddleChestScript

Short controlvar
Short button

Begin OnActivate
If ( controlvar == 0 )
MessageBox "Безголосый, но плачет; бескрылый, но парит; беззубый, но кусает;
безротый, но бормочет. Что это?", "Летучая мышь", "Старуха", "Ветер", "Привидение"
Set controlvar to 1
ElseIf ( controlvar > 1 )
Activate
EndIf
End

Begin GameMode
If ( controlvar == 1 )
Set button to GetButtonPressed
If ( button == -1 )
Return
ElseIf ( button == 2)
MessageBox "Правильный ответ."
Set controlvar to 2
Else
MessageBox "Неправильный ответ."
Cast Mg05FingerSpell15 Player
Set controlvar to 0
EndIf
ElseIf ( controlvar == 2 )
Activate
Set controlvar to 3
EndIf
End
```

Вот и все, ваш скрипт готов. Поздравляем! Если хотите, можете поэкспериментировать со скриптом, используя другие функции.

5.9 Как узнать больше.

После прочтения этого учебника вы можете спросить себя, как продолжить изучение написания скриптов? Хороший способ – посмотреть примеры из учебника или скрипты из игры (как написанные Bethesda, так и из модов). Попробуйте найти скрипт подобный тому, что вы хотите написать, затем скопируйте его и измените под ваши нужды. Прочитайте общую информацию на страницах функций и описания функций, которые могут понадобиться вам при осуществлении задуманного.

Классификация функций в функциональные типы должна помочь вам в поиске нужного. Наконец, официальные форумы - отличное место для поиска информации (используйте функцию поиска) или получения помощи в конкретной проблеме. А остальное – практика, практика и еще раз практика.

5.10 Последние строки.

Читатели, которые обратили внимание на функцию "If", должны были заметить, что ставить круглые скобки вокруг условий не обязательно. Я включил их в данный учебник, т.к. мне кажется, что это упорядочивает и упрощает понимание скрипта.

Наконец, объявляю огромную благодарность GhanBuriGhan за фантастическое "Руководство GhanBuriGhan’а по скриптам Morrowind для чайников" (GhanBuriGhan's excellent Morrowind Scripting for Dummies), в котором содержится прототип этого учебника. Я не смог связаться с ним, чтобы получить разрешение на его использование; если у него есть какие-либо вопросы насчет данного руководства, он вправе убрать или отредактировать его.

6. Функции в TES 4

6.1 Что такое функция? (Function)

Функции – это скриптовые операции, которые, в отличие от команд, напрямую взаимодействуют с игровым миром. Из всех скриптовых операций самое большое количество приходится именно на функции, которых на данный момент насчитывается 359 (в игре TES 4 Oblivion используется 353). Внушительное количество полезных функций – 96 - насчитывается в расширителе скриптов OBSE v0009.

Программисты, привыкшие к таким языкам программирования, как Pascal и Delphi, не найдут в скриптовом языке Обливион привычных процедур. Есть только функции.

Все без исключения функции возвращают какое-либо значение. Возвращаемые значения могут быть использованы при проверке условий "if" или же их можно сохранить в переменных, используя команду “set”.

Функции подразделяются на две подкатегории: пассивные и активные.
Пассивные функции проверяют определенные значения в игре и возвращают их численное значение. К примеру, GetActorValue возвращает определенное числовое значение, а GetDetected возвращает “1”, если цель обнаружена, или “0”, если нет и т.п.
Активные функции, в отличие от пассивных, вносят изменения в игровой мир и, как правило, возвращают логический результат (“1” или “0”) - были ли действия успешными или нет. RemoveSpell, к примеру, снимает заклинание с цели и возвращает “1”, если действие прошло успешно (т.к. на цели было заклятие). PlaceAtME создает какой-либо объект в локации вызвавшего эту функцию и возвращает ссылку на копию этого объекта.

6.2 Типы функций (Function Types)
Чтобы как-то систематизировать функции, в WIKI они были разделены на 22 категории:
AI Functions – Функции, связанные с искусственным интеллектом (Radiant AI)
Actor Functions – Функции, работающие с актерами.
Actor State Functions – Функции, связанные с состоянием актера.
Statistics Functions – Статистические функции, связанные с характеристиками актеров.
Actor Value Functions – Функции, модифицирующие характеристики актеров. Относятся также к статистическим функциям.
Animation Functions – Анимационные функции
Combat Functions – Функции боя
Condition Functions – Функции условий. Могут использоваться в качестве условий в любом месте редактора, где есть условия (стадии квеста, диалоги, пакеты, менеджер анимации и т.п.).
Crime Functions – Функции преступлений
Dialogue Functions – Функции диалогов
Faction Functions – Функции фракций
Magic Functions – Магические функции
Miscellaneous Functions – Прочие функции
Movement Functions – Функции движения
Object Functions – Объектные функции
Player Functions – Функции, обращающиеся только к игроку
Quest Functions – Квестовые функции
Reference Variable Functions – Функции, возвращающие указатели для установки ref- переменных
Time Functions – Функции времени.
Weather Functions – Функции погоды.
Console Functions – Консольные функции. По другому – консольные команды, а иногда – чит-коды или читы.
OBSE Function Types – Функции расширителя скриптового языка OBSE.

Многие из основных функций относятся сразу к нескольким категориям.
После некоторых размышлений мы решили, что оптимальным решением будет размещение описаний основных функций TES 4 CS в алфавитном порядке, отдельно будут описаны консольные функции и команды, а также новые функции расширителя скриптов OBSE.

6.3 Работа функций с копиями объектов (Reference functions)

Объектные функции

Большинство скриптовых функций являются объектными функциями, поскольку запускаются с копии объекта (Object reference). Локальные скрипты (скрипты на объектах и скрипты в поле result диалога) могут использовать неявный синтаксис обращения к копии.
Например:
CODE
GetDisposition player

GetDisposition – это объектная функция – отношение кого мы запрашиваем? Поскольку в этом случае мы не указываем копию объекта, подразумевается, что мы пишем:
CODE
thisReference.GetDisposition player; thisReference – копия, к которой прикреплен скрипт

Всегда можно использовать явный синтаксис, точно указывая копию, на которой вызывается функция. Нелокальные скрипты должны использовать явный синтаксис, поскольку они не запускаются с какой-либо копии. Например:
CODE
JauffreRef.GetDisposition player

Здесь мы запрашиваем отношение Джоффри(Jauffre) к игроку. Этот синтаксис может использоваться в любом скрипте, потому что мы явно указали копию объекта.
ПРИМЕЧАНИЕ: вы также можете запускать объектные функции с переменных типа ref, как если бы эти переменные сами были объектами.

Описание функций TES 4 Oblivion в алфавитном порядке:

A

Activate

Синтаксис:
CODE
Activate ActivatorID (optional), NormalActivationFlag (optional)

Примеры:
CODE
Activate player
Activate
Activate player, 1

Функция Activate указывает объекту выполнить его действие по-умолчанию.
Если ActivatorID опущен, то команда Activate будет использовать «текущий активатор» вызывающего объекта. Это очень полезно внутри блока OnActivate, когда вам нужно, чтобы активация проходила в обычном режиме, за исключением определенных условий.

Это означает, что вызов
CODE
Activate

будет эквивалентно
CODE
ref actingref
set actingref to GetActionRef
Activate actingref

Если используется ActivatorID, вызывающий объект исполнит действие, привязанное к ActivatorID.
CODE

Тип объекта	Активация
NPC	Диалог
Контейнер	Открыть
Дверь	Открыть
Оружие, броня, и т.д.	Поднять
Книга/Свиток	Читать

Если флаг NormalActivationFlag опущен, вызывающий объект исполняет нужное действие, пропуская любые блоки OnActivate, которые могут быть на нем. Обычно так и используется команда Activate, т.к. чаще всего она вызывается как раз внутри блока OnActivate.

Если NormalActivationFlag установлен в 1, вызывающий объект будет активирован обычным способом, включая блок OnActivate. В основном этот флаг аннулирует активацию по умолчанию как представлено выше. Используйте данную возможность осторожно — если вы вызовете Activate на сам объект изнутри блока OnActivate, вы тем самым запустите бесконечный цикл.

При осторожном использовании функция Activate может оказаться очень полезной для принудительного вызова и использования как псевдо-функция вызова между заскриптованным объектом и заклинаниями. Такое использование не предполагалось и требует тщательного планирования.

Пример 1:
Если вы вызовете этот скрипт на двери, то дверь будет действовать так, как будто ее активировал сам игрок (т.е., если это загрузочная дверь, игрок будет телепортирован в пункт назначения двери).
CODE
Activate player

Пример 2:
Чтобы не переключило блок OnActivate, оно также вызовет Activate, как будто скрипта и нет.
CODE
begin OnActivate
if MyCrazyCondition == 1
Activate
else
; делаем что-то еще
endif
end

Пример 3:
Этот скрипт запускает бесконечный цикл — как только заскриптованный объект активирован, блок OnActivate будет выполняться вечно.
CODE
float infinity
begin OnActivate
; НИКОГДА так не делайте!!
set infinity to infinity + .1
message "Infinity = %.1f", infinity
activate player 1
end

Пример 4:
Вот правильный способ использования NormalActivationFlag:
CODE
begin OnActivate
; активируем другой объект, когда этот активирован игроком
if IsActionRef player == 1
MyGate.Activate player 1
endif
end

Относится к типу: Object Functions

AddAchievement

Синтаксис:
CODE
AddAchievement AchievementNumber

Пример:
CODE
AddAchievement 12

Используется только в XBox!
Добавляет указанное достижение в профиль игрока. Т.к. достижения прописаны в движке, эта функция не используется в модостроении.
Относится к типу: ?????????

AddFlames

Синтаксис:
CODE
[ObjectID.]AddFlames

Добавляет объект пламени FlameNode к вызываемому объекту.



Пламя выключено ..... Пламя включено

См. также: FlameNode, CanHaveFlames, HasFlames, RemoveFlames
Относится к типу: Object Functions

AddItem

Синтаксис:
CODE
[ActorID|ContainerID.]AddItem ObjectID, Count

Пример:
CODE
AddItem MyObject, 1

CODE
Ref MyItem
Short count
set MyItem to ArenaAkaviriLongSword
set count to 1
player.additem MyItem Count

Добавляет число Count предметов ObjectID в инвентарь вызывающего контейнера.

Примечания:
В качестве ObjectID могут быть использованы переменные типа ref, а для Count - переменные типа short.
Данная функция не включит блок OnAdd потому, что предмет создается внутри инвентаря, а не добавляется в него. Для создания объектов из консоли, если вам нужно включить блок OnAdd, используйте функцию PlaceAtMe и поднимите предмет с земли.
Консольное использование:
Используя AddItem с консолью, вы должны использовать FormID желаемого предмета, а не его EditorID.
Поэтому для того, чтобы добавить себе отмычку, вместо команды
CODE
player.additem lockpick 1

вы должны набрать в консоли
CODE
player.additem 00000A 1
Kedrigh (wiki) добавил:
"FormID находится в CS и имеет вид: yyxxxxxx.
yy используется для ссылки на плагин, а xxxxxx - чтобы сослаться на элемент внутри плагина (и он не изменяется). Таким образом, если вы загружаете мастер-файл Oblivion в CS и, немного поработав, создали новый объект, то
FormID этого объекта будет выглядеть как 01xxxxxx.
Часть FormID - yy - изменяется в зависимости от загруженных модов, которые вы отметили при загрузке."

Прим. Garin:
yy зависит от порядкового номера загруженного плагина, которые при загрузке сортируются по дате.

См. также: RemoveItem, GetItemCount
Относится к типу: Object Functions | Actor Functions

AddScriptPackage

Синтаксис:
CODE
[ActorID.]AddScriptPackage PackageID

Пример:
CODE
AddScriptPackage MQ07BeggarToTavern

Описание:
Функция AddScriptPackage добавляет скриптовый пакет для вызывающего актера. Когда эта функция вызвана, этот пакет станет преобладающим над всеми другими пакетами.
Примечания:
Актер может иметь только один скриптовый пакет. При вызове этой функции дважды, второй пакет заменит первый.
Отметим, что в случае, когда добавленный пакет не принуждает актера к чему-то ("должен завершить" (must complete) и/или "должен достичь расположения" (must reach location)), он будет снят, когда актер в следующий раз переоценит этот пакет.
Справка:
Список скриптов Обливиона, которые используют AddScriptPackage (WIKI):
AmuseiScript
ArmandChristopheScript
BanditSentryFemaleScript
BanditSentryMaleScript
bernadettepenelesscript
CPRollingRock01SCRIPT
DASanguineSpell
DavideSurilieScript
HieronymusLexScript
KurtTestMagicEffect2
MartinScript
MethredhelScript
MQ09Script
MQ14StatueScript
MQ15AnaxesDoorSCRIPT
MS45DarMaScript
OrrinScript
TG03ChapelUndercroftGuardScript
TG11BlindMonkBossScript
ToutiusSextiusScript
TrigZoneP layerDoonce01DABoethiaSCRIPT01
TrigZonePlayerDoonce01DABoethiaSCRIPT02
TrigZonePlayerDoonce01DABoethiaSCRIPT03
TrigZonePlayerDoonce01DABo ethiaSCRIPT04
TrigZonePlayerDoonce01DABoethiaSCRIPT05
TrigZonePlayerDoonce01DABoethiaSCRIPT06
TrigZonePlayerDoonce01DABoethiaSCRIPT07
TrigZonePlayerDoonce01DABoethiaSCRIPT08
TrigZonePlayerDoonce01DABoethiaSCRIPT09
См. также: RemoveScriptPackage
Относится к типам: AI Functions | Actor Functions

AddSpell

Синтаксис:
CODE
[ActorID.]AddSpell SpellID

Пример:
CODE
AddSpell AbWeaknessNormalWeapons

Добавляет заклинание вызывающему актеру. Заклинаниями также считаются болезни, дополнительные способности, благословения (powers). Когда добавляется способность или болезнь, отображается соответствующий магический эффект. Когда добавляется заклинание или благословение, показывается скастованное заклинание\благословение. Есть и другие важные отличия, смотрите Spell, чтобы узнать подробности.

Примечания:
Addspell работает только с типами заклинаний, указанными в секции заклинаний редактора. Её нельзя использовать для заклинаний, добавленных с помощью скриптов. Для динамического добавления заклинания, основываясь на его уровне, используйте следующий скрипт в качестве руководства:
CODE
if (Player.GetLevel >= 25)
player.addspell InnerFire25
elseif (player.Getlevel >= 20)
player.addspell InnerFire20
elseif (player.Getlevel >= 15)
player.addspell InnerFire15
elseif (player.Getlevel >= 10)
player.addspell InnerFire10
elseif (player.Getlevel >= 5)
player.addspell InnerFire05
else
player.addspell InnerFire01
endif
Будьте осторожны при использовании данной функции на неуникальных актерах. Добавление способности или заклинания на актёра изменит базовый объект, и все актёры, созданные из данного базового объекта также подвергнуться изменению.
Как и большинство функций, эта имеет отличия при использовании в консоли - вместо SpellID надо указывать FormID.
Обратите внимание, что большинство заклинаний не работают как способности, включая "Перо", "Бремя" и "Дезинтеграцию", так же как и любые заклинания, заставляющие цель реагировать на заклинателя, например "Поднять мёртвого", "Очаровать" или "Деморализовать".
См. также: RemoveSpell
List of existing scripts that use AddSpell
Относится к типу: Magic Functions | Actor Functions

AddTopic

Синтаксис:
CODE
AddTopic TopicID

Пример:
CODE
AddTopic HiddenCave

Используйте эту функцию, чтобы добавить тему в список известных тем игрока.
Только темы из этого списка появляются в перечне тем NPC во время диалога. Темы также можно добавить, используя список AddTopic в окне редактора диалогов.
См. также: List of functions that use AddTopic (wiki)
Относится к типу: Dialogue Functions

AdvancePCLevel

Синтаксис:
CODE
AdvancePCLevel
advLevel

Ручное повышение уровня игрока, вызывающее меню «повышение уровня».
Относится к типу: Player Functions

AdvancePCSkill

Синтаксис:
CODE
AdvancePCSkill Skill Amount
advSkill Skill Amount

Пример:
CODE
AdvancePCSkill Blade 2

Функция AdvancePCSkill повышает навык игрока (Skill) на величину Amount, как если бы игрок получил его при многократном использовании в игре или у тренера.

Эта функция стала объектом спора в обсуждении ModPCSkill. Пожалуйста, прочтите эту страницу перед использованием данной функции.
Прим. zOmb (ZomBoss):
Тестовое дополнение:
AdvancePCSkill добавляет игроку очко скилла. Причём, если скилл в главных, то повышается и шкала повышения уровня.
Вызов функции с отрицательным аргументом приводит к зависанию игры.
Относится к типу: Player Functions

Autosave

Синтаксис:
CODE
Autosave

Вызов этой функции приводит к немедленному автосохранению. Запись производится в тот же слот памяти, что и обычное автосохранение.
Относится к типу: ?????????

C

CanHaveFlames

Синтаксис:
CODE
[objectID.]CanHaveFlames

Возвращает 1, если объект может использовать пламя (светильник, например)

См. также: FlameNode, HasFlames, AddFlames, RemoveFlames
Относится к типу: Object Functions | Condition Functions

CanPayCrimeGold

Синтаксис:
CODE
[ActorID.]CanPayCrimeGold

Возвращает 1, если актер имеет достаточное количество золота в инвентаре, чтобы оплатить штраф за преступление, и 0, если нет.

См. также: GetCrimeGold, ModCrimeGold, SetCrimeGold
Относится к типу: Crime Functions | Condition Functions

Cast

Синтаксис:
CODE
[ActorID|ActivatorID.]Cast SpellID [TargetRefID]

Пример:
CODE
Cast ShrineAkatoshSpeedMagicka Player

При вызове функции Cast вызывающий объект кастует заклинание SpellID на цель TargetRefID (и только заклинания).

Относится к типу: Magic Functions | Actor Functions

CloseCurrentOblivionGate

Синтаксис:
CODE
CloseCurrentOblivionGate iNoResetFlag (optional)

Когда вызывается функция CloseCurrentOblivionGate, то мир Обливиона, в котором находится игрок, сбрасывается. Он возвращается туда, откуда он вошел в Обливион и поворачивается лицом к воротам. Затем ворота сворачиваются и помечаются как уничтоженные (не используемые более). Другие актеры помещаются за ворота и связанный мир сбрасывается.

Другие актеры (не принадлежащие к данной локации) также помещаются за ворота и связанный мир сбрасывается.
Если опциональный флаг iNoResetFlag = 1, то мир Обливиона после закрытия врат не сбрасывается.

См. также: CloseOblivionGate
Относится к типу: Miscellaneous Functions

CloseOblivionGate

Синтаксис:
CODE
CloseOblivionGate iNoResetFlag (optional)

Референсная функция. Если вызывающий объект – врата Обливиона, они закрываются (более подробно - см. CloseCurrentOblivionGate).

См. также: CloseCurrentOblivionGate
Относится к типу: Miscellaneous Functions

CompleteQuest

Синтаксис:
CODE
CompleteQuest QuestID

Пример:
CODE
CompleteQuest SQ09

Помечает квест как законченный. Единственный реальный эффект состоит в том, что квест будет отображаться в разделе «Завершенные» списка квестов.

См. также: StartQuest, StopQuest
Относится к типу: Quest Functions

CreateFullActorCopy

Синтаксис:
CODE
[ActorID.]CreateFullActorCopy

Пример 1:
Функция CreateFullActorCopy создает копию указанного актера (ActorID, в данном случае игрока) и его базы:
CODE
player.CreateFullActorCopy

Пример 2:
Функция создает копию вызывающего актера и возвращает ID нового объекта, поэтому может использоваться для того, чтобы задать ref-переменную, как это сделано в этом примере:
CODE
set newRef to CreateFullActorCopy

Копия представляет собой полный дубликат оригинального актера (на котором "висит" данный скрипт), включая инвентарь, но со следующими оговорками:
Объекты инвентаря дублируются также, как и в случае вызова функции DuplicateAllItems
У копии нет пакетов ИИ или фракций.
См. также: DeleteFullActorCopy
Относится к типу: Miscellaneous Functions | Actor Functions

D

DeleteFullActorCopy

Синтаксис:
CODE
[refID.]DeleteFullActorCopy

Пример:
CODE

newRef.DeleteFullActorCopy

Функция DeleteFullActorCopy удаляет созданную копию актера (refID) полностью и его базовый объект.
Примечание: Работает как функция return — строки скрипта, следующие после вызова, выполняться не будут.
См. также: CreateFullActorCopy
Относится к типу: Miscellaneous Functions

Disable

Синтаксис:
CODE
[ObjectID|ActorID.]Disable

Функция Disable отключает вызывающий объект. Отключенные объекты не загружаются в мир, отключенные актеры не обрабатывают свой ИИ, но скрипты на отключенных объектах ВЫПОЛНЯЮТСЯ.
Функция Disable не удаляет объекты из игры, она просто прекращает их отрисовку. Не полагайтесь на Disable, чтобы увеличить производительность путём удаления объектов.
См. также: Enable, GetDisabled
Относится к типу: Miscellaneous Functions

DisableLinkedPathPoints

Синтаксис:
CODE
[ObjectID.]DisableLinkedPathPoints

Функция DisableLinkedPathPoints отключает все путевые узлы, привязанные к вызывающему объекту ObjectID. Это нужно, чтобы изменить нахождение пути на геометрии, изменяющейся во время игры, например, на разводных мостах (drawbridges), движущихся решётках (portcullises), разваливающихся опорах (crumbling bridges) и т.п.
Чтобы привязать узел пути к объекту, выделите его в редакторе и нажмите "R".
См. также: EnableLinkedPathPoints, Category: Path Grids
Относится к типу: Miscellaneous Functions

DisablePlayerControls

Синтаксис:
CODE
DisablePlayerControls

При вызове функции DisablePlayerControls игрок не может двигаться, ждать, активировать предметы или получать доступ к журналу.
Примечание: при этом игрок всё ещё может осматриваться вокруг.
См. также: EnablePlayerControls
Относится к типу: Player Functions

Dispel

Синтаксис:
CODE
[ActorID].Dispel MagicID

Примеры:
CODE
Dispel ShrineAkatoshSpeedMagicka

CODE
NPCRef.Dispel ShrineAkatoshSpeedMagicka

Функция Dispel удаляет указанное заклинание/магию (MagicID), наложенные на актера ActorID.
Примечание: Эта функция должна работать и работает как команда Return, если вы пытаетесь удалить заклинание в его текущем скрипте. Например, в следующем скрипте сообщение показано не будет:
CODE
scn EXSpellScript
Begin ScriptEffectFinish
player.dispel EXSpell
messagebox "Не будет виден!"
End

См. также: DispelAllSpells
Относится к типу: Magic Functions | Actor Functions

DispelAllSpells

Синтаксис:
CODE
[ActorID.]DispelAllSpells

Функция DispelAllSpells удаляет все текущие заклинания/магию, наложенные на актера (ActorID).
См. также: Dispel
Относится к типу: Magic Functions | Actor Functions

Drop

Синтаксис:
CODE
[ActorID.]Drop ObjectID Count

Пример:
CODE
Drop Torch02 2

При вызове функции Drop вызывающий актер ActorID выбрасывает указанный предмет(ы) (ObjectID) в мир к своим ногам в указанном количестве (Count).
Относится к типу: Object Functions | Actor Functions

DropMe

Синтаксис:
CODE
[ObjectID.]DropMe

При вызове функции DropMe из контейнера (инвентаря) выбрасывается вызывающий предмет ObjectID. Если его в контейнере нет, функция не имеет эффекта.
Примечания:
Эта функция работает как функция Return - строки скрипта, следующие за ней, выполнены НЕ будут (т.к. объект уничтожает cfv себя в процессе удаления из инвентаря).
В отличие от RemoveMe, функция не сохраняет уровни изношенности, заряда или скриптовые переменные предмета.
См. также: RemoveMe
Относится к типу: Object Functions

DuplicateAllItems

Синтаксис:
CODE
[ActorID|ContainerID.]DuplicateAllItems TargetContainerID

Пример:
CODE
DuplicateAllItems BlackBrugoRef

Функция DuplicateAllItems дублирует все предметы, находящиеся в вызывающем контейнере (ActorID|ContainerID), в целевом контейнере TargetContainerID. Заскриптованные объекты дублируются как новые, незаскриптованные (но в остальном идентичные) - как версии (копии) самих себя. Квестовые предметы также дублируются, так что используйте эту функцию осторожно. Многие квесты полагают, что квестовые предметы уникальны — создание их второй копии может привести к нежелательным последствиям.
См. также: CreateFullActorCopy
Относится к типу: Category: Items | Actor Functions

DuplicateNPCStats

Синтаксис:
CODE
[ActorID.]DuplicateNPCStats NPCToCopyFromID

Пример:
CODE

myClone.DuplicateNPCStats player

Функция DuplicateNPCStats дублирует все характеристики, которые имеются у указанного персонажа (NPCToCopyFromID), на вызывающего NPC (ActorID), включая следующее:
Класс (class)
Уровень (level)
Навыки и атрибуты (Skills and attributes)

См. также: CreateFullActorCopy
Относится к типу: Miscellaneous Functions

E

Enable

Синтаксис:
CODE
[ObjectID.]Enable

Функция Enable включает вызывающий объект ObjectID, заблокированный ранее функцией Disable.
Все объекты включены в игре по умолчанию, а это означает, что они рендерятся в игре и, если они актеры, то отрабатываются их пакеты AI.
См. также: Disable, GetDisabled
Относится к типу: Miscellaneous Functions

EnableFastTravel

Синтаксис:
CODE
EnableFastTravel flag

Функция EnableFastTravel включает или отключает быстрое путешествие.
Если флаг (flag) установлен в "1", тогда разрешено, если в "0" = заблокировано до тех пор, пока снова не будет включено этой функцией.
Относится к типу: Miscellaneous Functions

EnableLinkedPathPoints

Синтаксис:
CODE
[ActorID.]EnableLinkedPathPoints

Функция EnableLinkedPathPoints включает все путевые узлы (пафгриды), связанные с объектом вызова ActorID.
Чтобы привязать точку пути к объекту, выберите в конструкторе точку пути и нажмите клавишу "R".
См. также: DisableLinkedPathPoints, Category: Path Grids
Относится к типу: Miscellaneous Functions

EnablePlayerControls

Синтаксис:
CODE
EnablePlayerControls

Функция EnablePlayerControls включает управление игрока, отключенное ранее функцией DisablePlayerControls
См. также: DisablePlayerControls
Относится к типу: Player Functions

EquipItem

Синтаксис:
CODE
[ActorID.]EquipItem ObjectID NoUnequipFlag

Пример:
CODE
EquipItem FavoriteCuirass

player.EquipItem CursedHelm 1

Вызов функции EquipItem заставляет актера ActorID одеть объект ObjectID. Если флаг NoUnequipFlag = 1, актер (включая и персонажа игрока), не сможет снять объект.
См. также: UnequipItem
Относится к типу: Object Functions | Actor Functions

EssentialDeathReload

Синтаксис:
CODE
EssentialDeathReload "Death message"

Пример:
CODE
EssentialDeathReload "Вы проиграли. Жоффре убит. Надежды больше нет."

Функция EssentialDeathReload заставляет игру перезагрузиться (также, как и после смерти персонажа игрока). Сначала появляется сообщение, затем меню перезагрузки.
Относится к типу: Miscellaneous Functions

EvaluatePackage

Синтаксис:
CODE
[ActorID.]EvaluatePackage PackageID
evp PackageID

Используйте функцию EvaluatePackage на указанном актере (ActorID), чтобы вынудить его переоценить пакеты и выбрать указанный PackageID, который должен выполняться немедленно. Это может быть необходимо в тех случаях, когда в скрипте изменились какие-либо условия. Например, если вы хотите, чтобы актер активировал рычаг, когда какой-нибудь флаг в скрипте установится в состояние "True" - "истина". Для того, чтобы актер активировал его сразу же после этого события, вам нужно вызвать EvaluatePackage и заставить актера немедленно переоценить пакеты, чтобы не ждать в течение целого часа. Эта функция должна использоваться осмотрительно, поскольку это может занять длительное время в случае, если актер имеет длинный список пакетов и условий.
Относится к типу - AI Functions | Actor Functions

F

ForceActorValue

Синтаксис:
CODE
[ActorID.]ForceActorValue StatName value
[ActorID.]ForceAV StatName value

Параметры:
CODE
ActorID - ID персонажа, необязательный.
StatName - название изменяемой характеристики
Value - значение, которое будет присвоено

Пример:
CODE
ForceActorValue Strength 50

Функция ForceActorValue изменяет для указанного актера (ActorID) его текущую характеристику (StatName) до определенного значения (value) (не путайте с SetActorValue, которая изменяет БАЗОВОЕ значение характеристики). Измененная характеристика будет отображаться в игре красным цветом (damaged - повреждение) или зеленым (restored - восстановление), чтобы информировать игрока, что это временная модификация. Неизмененные параметры будет отображаться синим цветом.
Основы:
Когда вы (или игра) используете GetActorValue, вы получаете сумму базового значения актера плюс значения трех модификаторов - игрового, магического и скриптового:
1. Game Modifier используется для "постоянного" эффекта, типа Damage и Restore.
2. Magic Modifier используется для "временного", развеваемого магического эффекта, такого как Drain и Fortify.
3. Script Modifier используется для "временного" неразвеиваемого скриптового эффекта, такого как благословения и проклятия.

Функции ModActorValue и ForceActorValue изменяют только скриптовый модификатор. Изменения, сделанные этими функциями, не восстанавливаются автоматически внутриигровыми средствами, подобно естественному восстановлению здоровья или магии. Для этого вам нужно "уничтожить" их в скрипте.
В скриптах эти функции ведут себя так, как описано выше, но в консоли они ведут себя несколько иначе - в связи с тем, что они использовались Bethesda для целевого тестирования объектов. См. описание для функции ModActorValue:
CODE
ModActorValue health, (value - GetActorValue health)
Как и в случае с ModActorValue, при вызове ForceActorValue из скрипта изменения будут временными.
См. также: Stats List, GetActorValue, GetBaseActorValue, SetActorValue, ModActorValue
Относится к типу: Actor Value Functions | Statistics Functions | Actor Functions

ForceCloseOblivionGate

Синтаксис:
CODE
ForceCloseOblivionGate

Функция ForceCloseOblivionGate мгновенно закрывает врата Обливиона — работает также, как и функция CloseOblivionGate, но анимация закрытия врат не проигрывается.
См. также: CloseOblivionGate, CloseCurrentOblivionGate
Относится к типу: Miscellaneous Functions

ForceFlee

Синтаксис:
CODE
[ActorID.]ForceFlee
[ActorID.]ForceFlee CellID
[ActorID.]ForceFlee CellID, ObjectRefID

Примеры:
CODE
ForceFlee ChorrolTheGreyMare
ForceFlee ChorrolTheGreyMare, BongondRef

Вызов функции ForceFlee заставляет вызывающего актера (ActorID) убегать, независимо от того, сражается в данный момент актер или нет.
Если функция вызывается без параметров, актер будет выбирать сам, куда убегать и когда остановиться, т.е. так, как поступают обычные актеры.
Если функция вызывается с CellID, актер будет убегать, пока не достигнет указанной ячейки.
Если функция вызывается с ObjectRefID, актер будет убегать, пока не достигнет указанной цели (ObjectRefID) (в данном случае ячейка CellID будет игнорироваться, но для нормальной компиляции скрипта ее требуется обязательно указать).
Есть полезное побочное действие этой функции, заключающееся в том, что если вызвать ее без параметров на разговаривающем в данный момент актере, то он мгновенно прекратит разговор и оба актера немедленно возвратятся к своим текущим пакетам.
Относится к типу : AI Functions | Actor Functions

ForceTakeCover

Синтаксис:
CODE
[ActorID.]ForceTakeCover TargetID Duration

Примеры: ForceTakeCover player 30
Функция ForceTakeCover вынуждает вызывающего актера (ActorID) отделиться от цели TargetID на указанное в секундах время (Duration).
Относится к типу: AI Functions | Actor Functions

ForceWeather

Описание: Функция ускоренной смены погоды
Тип: nil
Синтаксис:
CODE
ForceWeather WeatherID WeatherOverrideFlag (optional)

Параметры:
CODE

WeatherID - ID объекта погоды
WeatherOverrideFlag - необязательный, указывает, зафиксировать погоду или нет

Примеры:
CODE
ForceWeather Clear
ForceWeather OblivionStorm 1
fw SigilWhiteOut

Функция ForceWeather позволяет установить указанную погоду (WeatherID) немедленно после вызова. Если необязательный флаг WeatherOverrideFlag указан и установлен в "1", то погода изменяться не будет, пока не будет вызвана функция ReleaseWeatherOverride, в противном случае погода вернется к своему нормальному виду согласно текущей системы погоды в данной местности.
Тип погоды можно изменять с помощью двух функций - ForceWeather и SetWeather. Отличаются они только тем, что в первом случае переход произойдёт мгновенно, а во втором - плавно, в соответствии с настройками объекта погоды.
ID погоды (Console IDs):

Tamriel
CamoranWeather 000370CE
Clear 00038EEE
Cloudy 00038EFE
DefaultWeather 0000015E
Fog 00038EEF
Overcast 00038EEC
Rain 00038EF2
Snow 00038EED
Thunderstorm 00038EF1
OblivionStormTamriel 000836D5
OblivionStormTamrielMQ16 0006BC8B
Oblivion
OblivionDefault 00032E15
OblivionElectrical 00067198
OblivionMountainFog 000671A1
OblivionSigil 000C0999
OblivionStormOblivion 00067199
SigilWhiteOut 000C42DE
Other
MS14Sky 0018BCCF
См. также: ReleaseWeatherOverride, SetWeather
Относится к типу: Weather Functions

G

GetActionRef

Синтаксис:
CODE
set refVar to GetActionRef

Функция GetActionRef возвращает ref-переменную объекта, который в данный момент взаимодействует с заскриптованным объектом. Доступно только тогда, когда объект активируется или включается. Функция возвращает значение только в одном фрейме сразу после взаимодействия с объектом. Это означает, что она в основном будет полезна внутри блоков OnActivate или OnTrigger.
В некоторых случаях, если заскриптованный объект взаимодействует одновременно с несколькими объектами (например, в зоне действия объекта находится несколько переключателей), функция возвратит только самый последний из объектов взаимодействия.

См. также: IsActionRef
Относится к типу: Reference Variable Functions

GetActorValue

Краткое описание: Возвращает текущее значение характеристики.
Синтаксис:
CODE
[ActorID.]GetActorValue StatName
[ActorID.]GetAV StatName

Параметры:
CODE
ActorID - ID персонажа, необязательный
StatName - название характеристики

Пример:
CODE
GetActorValue Strength

Функция GetActorValue возвращает текущее модифицированное значение характеристики (StatName) для указанного актера ActorID. GetActorValue может использоваться с любыми характеристиками, доступными персонажу игрока или актеру (NPC или существо).

См. также: Stats List, GetBaseActorValue, ModActorValue, SetActorValue, ForceActorValue
Относится к типу: Actor Functions | Actor Value Functions | Statistics Functions | Condition Functions

GetAlarmed

Синтаксис:
CODE
[ActorID.]GetAlarmed

Функция GetAlarmed возвращает истину (1), если актер (ActorID) поднял тревогу.

Относится к типу: Crime Functions | Condition Functions | Actor Functions

GetAmountSoldStolen

Синтаксис:
CODE
GetAmountSoldStolen

Функция GetAmountSoldStolen возвращает полную сумму украденных товаров, которые игрок продал. Отметьте, что это количество золота, которое получил игрок по факту, а не базовая стоимость предметов.

См. также: ModAmountSoldStolen
Относится к типу: Player Functions | Condition Functions

GetAngle

Синтаксис:
CODE
[ActorID.]GetAngle axis

Пример:
CODE
GetAngle Z

Функция GetAngle возвращает текущий угол поворота объекта (ActorID) по выбранной оси (axis) (X, Y, или Z) в формате float относительно мировых координат.
Особые замечания (если функция вызвана на персонаже игрока):
Угол Z возвращается со значением от 0 до 360 град в зависимости от текущего положения головы персонажа.
Угол X возвращается (теоретически) в диапазоне от -90 до +90 градусов. В действительности игровой диапазон составляет -89....+89. Если игрок смотрит вверх, то предельное возвращаемое значение угла будет составлять -89 градусов, если же он смотрит вниз, на ноги, то предельное значение будет составлять +89.
Относится к типу: Movement Functions | Condition Functions

GetArmorRating

Краткое описание: Функция возвращает общий рейтинг брони.
Синтаксис:
CODE
[ActorID.]GetArmorRating

Параметры:
CODE
ActorID - ID персонажа, необязательный.

Функция GetArmorRating возвращает полный суммарный рейтинг всех доспехов, одетых в данный момент на указанного актера (ActorID).

Относится к типу: Statistics Functions | Condition Functions | Actor Functions

GetArmorRatingUpperBody

Синтаксис:
CODE
[ActorID.]GetArmorRatingUpperBody

Функция GetArmorRatingUpperBody не дает рейтинг брони (доспехов), но возвращает, какой тип (класс) брони носит актер в слоте верхней части тела.
Параметры: ActorID - ID персонажа, броню которого нужно проверить. Необязательный.
Возможные возвращаемые значения класса брони:
0 = нет брони
1 = легкая броня
2 = тяжелая броня

См. также: GetArmorRating, GetObjectValue(OBSE)
Относится к типу: Statistics Functions | Condition Functions | Actor Functions

GetAttacked

Синтаксис:
CODE
[ActorID.]GetAttacked

Функция GetAttacked возвращает 1, если вызывающий актер (ActorID) был атакован. Это значение сбрасывается, когда актер выходит из боя.
Примечание:
Не ясно, чем она отличается от IsInCombat.
Не ясно, работает ли эта функция вообще...

См. также: IsInCombat
Относится к типу: Combat Functions | Condition Functions | Actor Functions

GetBarterGold

Синтаксис:
CODE
GetBarterGold

Тип: целое, неотрицательное
Функция GetBarterGold возвращает доступное для обмена количество денег (золота) у NPC. Фактически, это максимальная цена, за которую NPC может что-либо купить.
См. также: ModBarterGold
Относится к типу: Statistics Functions | Condition Functions

GetBaseActorValue

Синтаксис:
CODE
[ActorID.]GetBaseActorValue StatName
[ActorID.]GetBaseAV StatName

Пример:
CODE
GetBaseActorValue Health

Параметры:
ActorID - ID персонажа, необязательный
StatName - название характеристики
Описание: Функция GetBaseActorValue возвращает базовое значение указанной характеристики StatName у указанного актера ActorID. Чтобы узнать текущее (модифицированное) значение, используйте функцию GetActorValue.
См. также: Stats List, GetActorValue
Относится к типу: Actor Functions | Actor Value Functions | Statistics Functions | Condition Functions

GetButtonPressed

Синтаксис:
CODE
GetButtonPressed

Пример:
CODE
set buttonVar to GetButtonPressed

При первом вызове функции GetButtonPressed, после того, как в игре была нажата кнопка в диалоговом окошке, созданном функцией MessageBox, GetButtonPressed возвратит номер нажатой кнопки ("0" означает первую кнопку, "1" - вторую, и т.д.). Во всех остальных случаях (если кнопка в предыдущих фреймах не нажималась либо значение нажатой кнопки уже было считано и сохранено в переменной) будет возвращаться "-1".
Отметьте, что функция реагирует на MessageBox, созданный в этом же скрипте.
Пример скрипта:
CODE
Begin OnActivate
if IsActionRef player == 1
messagebox "У вас 3 варианта:", "Вариант 1", "Вариант 2", "Вариант 3"
endif
end

begin gamemode
set button to getbuttonpressed
if button > -1
if button == 0
; Вариант 1
elseif button == 1
; Вариант 2
elseif button == 2
; Вариант 3
endif
endif
end

См. также: MessageBox
Относится к типу: Miscellaneous Functions

GetClassDefaultMatch

Синтаксис:
CODE
GetClassDefaultMatch

Функция GetClassDefaultMatch возвращает 0, 1 или 2 в зависимости от того, насколько текущий класс персонажа игрока соответствует "классу по-умолчанию":
CODE
2 = соответствует полностью
1 = совпадает специализация (Сражение, Магия или Скрытность)
0 = нет совпадений
Например, если класс игрока по-умолчанию — Скаут (Scout), а текущий класс — Варвар (Barbarian), то эта функция вернет 1. Если класс игрока по-умолчанию — Целитель (Healer), а текущий класс — Варвар (Barbarian), то эта функция возвратит 0 (полностью различные специализации).

См. также: GetIsClassDefault, SetInCharGen
Относится к типу: Player Functions | Condition Functions

GetClothingValue

Синтаксис:
CODE
[ActorID.]GetClothingValue

Функция GetClothingValue возвращает «стоимость одеяния» одетой в данный момент на вызывающем актере (ActorID) одежды и брони, в интервале от 0 до 100. Ценность каждого предмета масштабируется в зависимости от процента покрытия

данного слота (поэтому дорогая роба будет иметь больший вес, чем дорогие ботинки). Эту функцию лучше всего использовать для диалогов, чтобы NPC реагировали на то, как богато или бедно одет кто-либо.
Следующие игровые настройки используются, чтобы определить ценность одежды, и возвращаются функцией GetClothingValue. Драгоценности, броня и одежда регулируются независимо:
fClothingArmorBase
Это базовое значение стоимости для ношения брони любого типа.
Значение по умолчанию: 0.000
fClothingArmorScale
Это множитель для денежной стоимости носимой брони.
Значение по умолчанию: 0.2500
fClothingBase
Это базовая стоимость для ношения повседневных одежд любого типа.
Значение по умолчанию: 0.000
fClothingClassScale
Это множитель для денежной стоимости всей носимой одежды.
Значение по умолчанию: 3.000
fClothingJewelryBase
Это базовая стоимость для ношения драгоценности любого типа.
Значение по умолчанию: 5.000
fClothingJewelryScale
Это множитель для денежной стоимости любой носимой драгоценности.
Значение по умолчанию: 0.1000
Относится к типу: Actor State Functions | Condition Functions

GetCombatTarget

Синтаксис:
CODE
[ActorID.]GetCombatTarget

Пример:
CODE
set refVar to GetCombatTarget

Функция GetCombatTarget возвращает текущую боевую цель вызывающего актера ActorID.
Но если вызывать эту функцию с явным указанием персонажа игрока -Player.GetCombatTarget, функция всегда будет возвращать 0, так как игра не может определить, какую цель выбрал игрок.

Относится к типу: Reference Variable Functions | Actor Functions

GetContainer

Синтаксис:
CODE
set refVar to GetContainer

Функция GetContainer возвращает ref-переменную (ссылку) на вызывающий объектный контейнер (если таковой имеется).

Относится к типу: Reference Variable Functions

GetCrime

Синтаксис:
CODE
[ActorID.]GetCrime CriminalActorID [CrimeType]

Пример:
CODE
GetCrime player, 3

Функция GetCrime возвращает 1, если актер CriminalActorID совершил определенный тип преступления [CrimeType] против вызывающего актера ActorID И вызывающий актер знает об этом. Если тип преступления [crime type] опущен, то истина возвращается, если вызывающий знает о любом преступлении, совершенном актером CriminalActorID против него.
Типы преступлений [crime type]:
CODE
0 = Кража
1 = Карманная кража
2 = Нарушение территории

3 = Атака
4 = Убийство

См. также: Crime Types
Относится к типу: Crime Functions | Condition Functions | Actor Functions

GetCrimeGold

Синтаксис:
CODE
[ActorID.]GetCrimeGold

Функция GetCrimeGold возвращает текущее значение "криминального" золота у актера ActorID. «Криминальное Золото» ("Crime Gold") — это текущий штраф, который актер заработал на преступлениях.

См. также: ModCrimeGold, SetCrimeGold
Относится к типу: Crime Functions | Condition Functions

GetCrimeKnown

Синтаксис:
CODE
[ActorID.]GetCrimeKnown CrimeType, Criminal, Victim

Пример:
CODE
GetCrimeKnown 4, EvilJoe, PoorBob

Функция GetCrimeKnown возвращает 1, если вызывающий актер (ActorID) знает об указанном преступлении (CrimeType).

См. также: Crime Types
Относится к типу: Crime Functions | Actor Functions

GetCurrentAIPackage

Синтаксис:
CODE
[ActorID.]GetCurrentAIPackage

Функция GetCurrentAIPackage возвращает следующие значения для типов пакетов AI, имеющихся у вызывающего актера ActorID:
CODE
0 = EXPLORE
1 = FOLLOW
2 = ESCORT
3 = EAT
4 = SLEEP
5 = COMBAT
6 = DIALOGUE
7 = ACTIVATE
8 = ALARM
9 = SPECTATOR
10 = FLEE
11 = TRESPASS
12 = GREET
13 = WANDER
14 = TRAVEL
15 = ACCOMPANY
16 = USEITEMAT
17 = AMBUSH
18 = FLEE_NON_COMBAT
19 = CAST_MAGIC
20 = COMBAT_LOW
21 = GET_UP
22 = MOUNT_HORSE
23 = DISMOUNT_HORSE
24 = DO_NOTHING
25 = CAST_TARGET_SPELL

26 = CAST_TOUCH_SPELL
27 = VAMPIRE_FEED
28 = SURFACE
29 = SEARCH_FOR_ATTACKER
30 = CLEAR_MOUNT_POSITION
31 = SUMMON_CREATURE_DEFEND
32 = MOVEMENT_BLOCKED
См. также: GetCurrentAIProcedure
Относится к типу: AI Functions | Condition Functions | Actor Functions

GetCurrentAIProcedure

Синтаксис:
CODE
[ActorID.]GetCurrentAIProcedure

Функция GetCurrentAIPackage возвращает следующие значения для типов AI-процедур, имеющихся у вызывающего актера ActorID:
CODE
0 = TRAVEL
1 = ACTIVATE
2 = AQUIRE
3 = WAIT
4 = DIALOGUE
5 = GREET
6 = GREET_DEAD
7 = WANDER
8 = SLEEP
9 = OBSERVE_COMBAT
10 = EAT
11 = FOLLOW
12 = ESCORT
13 = COMBAT
14 = ALARM
15 = PURSUE
16 = FLEE
17 = DONE
18 = YIELD
19 = TRAVEL_TARGET
20 = CREATE_FOLLOW
21 = GET_UP
22 = MOUNT_HORSE
23 = DISMOUNT_HORSE
24 = DO_NOTHING
25 = CAST_SPELL
26 = AIM
27 = ACCOMPANY
28 = USE_ITEM_AT
29 = VAMPIRE_FEED
30 = WAIT_AMBUSH
31 = SURFACE
32 = WAIT_FOR_SPELL
33 = CHOOSE_CAST
34 = FLEE_NON_COMBAT
35 = REMOVE_WORN_ITEMS
36 = SEARCH
37 = CLEAR_MOUNT_POSITION
38 = SUMMON_DEFEND
39 = MOVEMENT_BLOCKED_WAIT

Отметим, что процедура и текущий пакет - суть не одно и то же. Отдельный пакет может содержать различные возможные процедуры - в зависимости от того, что делает в настоящий момент актер. Например, в пакете "Travel" актер будет выполнять процедуру путешествия - "Travel" (когда актер перемещается к цели назначения), а затем, прибыв в место назначения, он будет выполнять процедуру ожидания - "Wait".
Процедура может также и не быть связанной с лежащим в ее основе пакетом - какое-нибудь незапланированное событие может заставить актера начать новую процедуру без изменения пакета.
Смотрите также: GetCurrentAIPackage
Относится к типу: AI Functions | Condition Functions | Actor Functions

GetCurrentTime

Синтаксис:
CODE
GetCurrentTime

Тип возвращаемого значения: float (вещественное)
Параметры: нет
Описание: Функция GetCurrentTime возвращает текущее значение игрового времени в десятичном формате с плавающей запятой.
Примеры:
CODE
GetCurrentTime

1) Текущее игровое время = 4:30 am (4:30), возвращаемое значение - 4.5
2) Текущее игровое время = 7:45 pm (19:45), возвращаемое значение - 19.75
Относится к типу: Time Functions

GetCurrentWeatherPercent

Синтаксис:
CODE
GetCurrentWeatherPercent

Функция GetCurrentWeatherPercent возвращает информацию о том, на какой стадии находится процесс смены типа погоды в текущей локации.
Любое значение в интервале от 0 до 1 говорит о том, что погода меняется.
Тип: вещественное, в диапазоне от 0 до 1
Относится к типу: Weather Functions | Condition Functions

GetDayOfWeek

Синтаксис:
CODE
GetDayOfWeek

Описание: Функция GetDayOfWeek возвращает текущее значение дня недели.
Тип возвращаемого значения: short (целочисленное короткое)
Параметры: нет
Возвращаемые значения:

Возвр.знач. - Игров. день недели - День недели
---
0 - Sundas - воскресенье
1 - Morndas - понедельник
2 - Tirdas - вторник
3 - Middas - среда
4 - Turdas - четверг
5 - Fredas - пятница
6 - Loredas - суббота

Относится к типу: Time Functions

GetDead

Синтаксис:
CODE
[ActorID.]GetDead

Пример:
CODE
BossGoblin.GetDead

Функция GetDead возвращает истину, если вызывающий объект ActorID мертв.
См. также: GetDeadCount
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetDeadCount

Синтаксис:
CODE
GetDeadCount ActorBaseType

Пример:
CODE
GetDeadCount Goblin

Функция GetDeadCount возвращает общее количество убитых существ/NPC указанного типа (ActorBaseType). Заметьте, что параметр ActorBaseType — это базовый тип актеров, например, гоблины или спригганы, поэтому вы не сможете использовать эту функцию, чтобы проверить, был ли убит конкретный объект, указав его ActorID.
См. также: GetDead
Относится к типу: Actor State Functions | Condition Functions

GetDestroyed

Синтаксис:
CODE
[ActorID.]GetDestroyed

Функция GetDestroyed возвращает 1, когда вызывающий актер (ActorID) помечен на уничтожение, и 0 — во всех других случаях.
Относится к типу: Miscellaneous Functions | Condition Functions

GetDetected

Синтаксис:
CODE
[ActorID.]GetDetected TargetID

Пример:
CODE
GetDetected player

Функция GetDetected возвращает 1, если вызывающий актер (ActorID) в данный момент заметил указанную цель - TargetID.
См. также: IsActorDetected, GetDetectionLevel, Category:Detection
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetDetectionLevel

Синтаксис:
CODE
[ActorID.]GetDetectionLevel TargetActor

Пример:
CODE
ScaryGuard.GetDetectionLevel player

Функция GetDetectionLevel используется для проверки текущего уровня обнаружения вызывающим актером (ActorID) указанной цели (TargetActor).
Возможны следующие уровни обнаружения:
CODE
0 = Потерян (Lost)
1 = Не виден (Unseen)
2 = Замечен (Noticed)
3 = Виден (Seen)

См. также: GetDetected, IsActorDetected, Category:Detection
Относится к типу: Crime Functions | Condition Functions | Actor Functions

GetDisabled

Синтаксис:
CODE
[ActorID.]GetDisabled

Функция GetDisabled возвращает 1, если вызывающий актер (ActorID) отключен.
См. также: Enable, Disable
Относится к типу: Object Functions | Condition Functions

GetDisease

Синтаксис:
CODE
[ActorID.]GetDisease

Функция GetDisease возвращает 1, если вызывающий актер (ActorID) заражен.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetDisposition

Синтаксис:
CODE
[ActorID.]GetDisposition TargetActor

Пример:
CODE
GetDisposition Baurus

Параметры:
ActorID - ID вызывающего персонажа, необязательный
TargetActor - указанный ID целевого актера
Функция GetDisposition возвращает текущее отношение вызывающего актера (ActorID) к указанному целевому актеру (TargetActor). Учитываются все модификаторы, в данный момент влияющие на отношение.
Относится к типу: Statistics Functions | Condition Functions | Actor Functions

GetDistance

Синтаксис:
CODE
[ObjectID1.]GetDistance ObjectID2

Функция GetDistance возвращает расстояние в игровых единицах длины от вызывающего объекта (ObjectID1) до указанного явно объекта "ObjectID2".
Игровые единицы игры Обливион соответствуют таковым для Morrowind.
К примеру, длина внешней ячейки равна 4096 единицам (units).
Подробнее о единицах длины см. здесь: Oblivion Units
Примеры:
CODE
GetDistance TestMarkerRef

CODE
Set distance to Getdistance Player

CODE
Ref Actor
Begin OnActivate
set Actor to GetActionRef
if Getdistance Actor > 70
messagebox "Подойдите поближе, мой друг!"
endif
end

Относится к типу: Movement Functions | Condition Functions

GetDoorDefaultOpen

Синтаксис:
CODE
[DoorID.]GetDoorDefaultOpen

Функция GetDoorDefaultOpen возвращает 1, если дверь (DoorID) открыта по-умолчанию, и 0 — если по-умолчанию закрыта.

Состояние по-умолчанию — это состояние двери, в которое она вернется после того, как ячейка выгрузится из памяти и игрок вновь войдет в нее, вне зависимости от того, в каком состоянии она осталась перед этим. Состояние по-умолчанию может быть задано через редактор в окне редактирования свойств объекта, или через скрипт командой SetDoorDefaultOpen.
Относится к типу: Miscellaneous Functions | Condition Functions

GetEquipped

Синтаксис:
CODE
[ActorID.]GetEquipped ObjectID

Функция GetEquipped возвращает 1, если вызывающий актер (ActorID) одет в одежду ObjectID.
Тип возвращаемой переменной - long (целочисленная длинная)
См. также: OnEquip, OnActorEquip
Относится к типу: Object Functions | Condition Functions | Actor Functions

GetFactionRank

Синтаксис:
CODE
[ActorID.]GetFactionRank FactionID

Пример:
CODE
GetFactionRank FightersGuild

Функция GetFactionRank возвращает ранг вызывающего актера (ActorID) во фракции FactionID. Если актер не числится в указанной фракции, функция возвратит "-1".
См. также: GetFactionRankDifference, GetInFaction, ModFactionRank, SetFactionRank
Относится к типу: Faction Functions | Condition Functions | Actor Functions

GetFactionRankDifference

Синтаксис:
CODE
[ActorID.]GetFactionRankDifference FactionID TargetActor

Пример:
CODE
GetFactionRankDifference FightersGuild JoeFightersGuildRef

Функция GetFactionRankDifference возвращает разницу в рангах между вызывающим актером (ActorID) и указанным целевым актером (TargetActor) в указанной фракции FactionID.
Относится к типу: Faction Functions | Condition Functions | Actor Functions

GetFactionReaction

Синтаксис:
CODE
GetFactionReaction FactionID1 FactionID2

Пример:
CODE
GetFactionReaction FightersGuild BlackwoodCompany

Функция GetFactionReaction возвращает реакцию одной фракции (FactionID1) по отношению ко второй (FactionID2).
Относится к типу: Faction Functions | Condition Functions

GetFatiguePercentage

Синтаксис:
CODE
[ActorID.]GetFatiguePercentage

Функция GetFatiguePercentage возвращает текущую усталость вызывающего актера (ActorID), как долю от базового показателя в диапазоне (0.0 ... 1.0).
Тип: Вещественное, в диапазоне от 0 до 1
Параметры: ActorID - ID персонажа, необязательный

Относится к типу: Statistics Functions | Condition Functions | Actor Functions

GetForceRun

Синтаксис:
CODE
[ActorID.]GetForceRun

Функция GetForceRun возвращает 1, если актер (ActorID) в данный момент находится в состоянии бега.
См. также: SetForceRun
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetForceSneak

Синтаксис:
CODE
[ActorID.]GetForceSneak

Функция GetForceSneak возвращает 1, если вызывающий актер (ActorID) находится в состоянии скрытности.
См. также: SetForceSneak
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetFriendHit

Синтаксис:
CODE
[ActorID.]GetFriendHit TargetActor

Пример:
CODE
FriendlyGuard.GetFriendHit player

Функция GetFriendHit возвращает число «дружеских» ударов, нанесенных вызывающему актеру (ActorID) указанным актером TargetActor.
Относится к типу: Combat Functions | Condition Functions | Actor Functions

GetFurnitureMarkerID

Синтаксис:
CODE
GetFurnitureMarkerID

Функция GetFurnitureMarkerID возвращает 0 для сломанных вещей, 1...10 - для кроватей и 11...20 для стульев. Это узлы маркеров (FurnitureMarker), встроенные в объекты мебели, которую актер может использовать. Эта функция используется менеджером анимации (idle manager), чтобы выбрать нужную анимацию для начала или конца использования мебели. Например, когда актер садится на стул, функция возвратит его тип, чтобы он проиграл анимация корректно.
Относится к типу: Object Functions | Condition Functions

GetGameSetting

Синтаксис:
CODE
GetGameSetting GameSettingName
getGS GameSettingName

Примеры:
CODE
GetGameSetting fBribeBase

Функция GetGameSetting возвращает значение указанной игровой настройки GameSettingName.
См. также: Con_SetGameSetting (OBSE)
Относится к типу: Miscellaneous Functions

GetGlobalValue

Синтаксис:
CODE
GetGlobalValue GlobalVariableName

Пример:
CODE
GetGlobalValue GameDaysPassed

Функция GetGlobalValue возвращает значение указанной глобальной переменной GlobalVariableName.
Примечание: Доступна только как функция условия в диалоге. В обычных скриптах можно просто использовать имя глобальной переменной:
CODE
set myVar to GameDaysPassed

См. также: GetQuestVariable, GetScriptVariable
Относится к типу: Miscellaneous Functions | Condition Functions

GetGold

Синтаксис:
CODE
[ActorID.]GetGold

Параметры:
ActorID - ID персонажа, необязательный.
Функция GetGold возвращает общее количество золота у указанного актера (ActorID).
Относится к типу: Statistics Functions | Condition Functions | Actor Functions

GetHeadingAngle

Синтаксис:
CODE
[ActorID.]GetHeadingAngle ObjectRefID

Пример:
CODE
GetHeadingAngle StrangeStatue

Функция GetHeadingAngle возвращает угол поворота между вызывающим объектом (ActorID) и указанным явно объектом ObjectRefID в пределах от -180 до 180.
Например, игрок смотрит на север. Объект StrangeStatue находится к западу от игрока. Вызов следующей команды вернет "-90" градусов:
CODE
player.GetHeadingAngle StrangeStatue

Если добавить это значение к текущему повороту игрока, он повернется лицом к статуе:
CODE
set angle to player.getangle z + player.GetHeadingAngle StrangeStatue
player.setangle z angle

Относится к типу: Movement Functions | Condition Functions

GetIdleDoneOnce

Синтаксис:
CODE
GetIdleDoneOnce

Функция GetIdleDoneOnce используется в менеджере анимации (idle manager), чтобы пометить определенное анимационное движение (idle) для одиночного проигрывания в конкретной ситуации.
Например, если вы вставите анимацию приветствия (salute idle) в приветственный диалог, происходящий между двумя NPC, и выставите в диалоге
CODE
GetIdleDoneOnce == 1
то в игре NPC выполнят ее всего один раз, а затем будут пропускать ее до конца диалога.
Относится к типу: Miscellaneous Functions | Condition Functions

GetIgnoreFriendlyHits

Синтаксис:
CODE

[ActorID.]GetIgnoreFriendlyHits

Функция GetIgnoreFriendlyHits возвращает 1, если вызывающий актер (ActorID) в данный момент игнорирует «дружеские» удары.
См. также: SetIgnoreFriendlyHits
Относится к типу: Combat Functions | Condition Functions | Actor Functions

GetInCell

Синтаксис:
CODE
[ActorID.]GetInCell CellNameID

Пример:
CODE
GetInCell Chorrol

Функция GetInCell возвратит истину, если выражение "CellNameID" совпадает с первыми буквами ID названия текущей ячейки, в которой находится вызывающий актер (ActorID). Подчеркнем еще раз, что совпадение должно быть полным и именно первых букв. Если название совпадает с центральными частями ID ячейки или с выражением в конце, то будет возвращено False.
В приведенном примере при вызове GetInCell Chorrol функция возвратит истину, если вызывающий персонаж, на котором "висит" этот скрипт, находится в ячейках Chorrol, ChorrolArena или ChorrolFarm и подобных. Но если вы вызовете эту же функцию с словосочетанием "Farm" - GetInCell Farm, то возвращаемый результат окажется ложным (False), даже когда персонаж находится в ячейке ChorrolFarm. Таким образом вы никогда не обнаружите эту ячейку.
Примечание: отметьте, что CellNameID верно только для внутренних ячеек - интерьеров. Для внешних ячеек (экстерьеров) эта функция не работает. Иногда необходимо создать «пустую» внутреннюю ячейку (например, Chorrol), чтобы использовать GetInCell для создания набора взаимосвязанных внутренних и внешних ячеек (например, все ячейки ChorrolXXX).

Перечень всех ячеек Обливиона, отсортированные по имени (а не по ID!) в алфавитном порядке и с указанием их ID, вы можете найти на сайте IGN на страницеThe Elder Scrolls IV: Oblivion Map Warping FAQ :
http://faqs.ign.com/articles/697/697909p1.html
Напомним, что как раз вводить в качестве CellNameID следует именно первую часть названия ID интерьера.
См. также: GetInCellParam, GetInSameCell, GetInWorldspace
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetInCellParam

Синтаксис:
CODE
GetInCellParam CellNameID ActorID

Пример:
CODE
GetInCellParam Chorrol JauffreRef

Функция GetInCellParam возвратит истину, если выражение "CellNameID" совпадает с первыми буквами ID названия текущей ячейки, в которой находится указанный в виде параметра актер (ActorID). Подчеркнем еще раз, что совпадение должно быть полным и именно первых букв. Если название совпадает с центральными частями ID ячейки или с выражением в конце, то будет возвращено False.
В функции GetInCellParam реализована та же функциональность, что и в GetInCell, кроме того, что эта функция не запускается на объекте — актер указывается в качестве параметра (более подробно смотрите в описании GetInCell)
См. также: GetInCell, GetInSameCell
Относится к типу: Actor State Functions | Condition Functions

GetInFaction

Синтаксис:
CODE
[ActorID.]GetInFaction FactionID

Пример:
CODE
GetInFaction MagesGuild

Функция GetInFaction возвращает 1, если вызывающий актер (ActorID) состоит в указанной в качестве параметра фракции (FactionID).

См. также: GetFactionRank
Относится к типу: Faction Functions | Condition Functions | Actor Functions

GetInSameCell

Синтаксис:
CODE
[ObjectID|ActorID.]GetInSameCell TargetRefID

Пример:
CODE
GetInSameCell player

Функция GetInSameCell возвращает истину (1), если вызывающий объект находится в той же ячейке, что и указанный в качестве параметра TargetRefID.
См. также: GetInCell, GetInCellParam
Относится к типу: Actor State Functions | Condition Functions

GetInWorldSpace

Синтаксис:
CODE
[ActorID.]GetInWorldspace WorldspaceName

Пример:
CODE
GetInWorldspace ChorrolWorld

Функция GetInWorldSpace возвращает 1, если вызывающий актер (ActorID) находится в одной из внешних ячеек (WorldspaceName) определенного пространства игрового мира. Всегда возвращает 0 во внутренней ячейке.
См. также: GetInCell
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetInvestmentGold

Синтаксис:
CODE
[ActorID.]GetInvestmentGold

Параметры:
ActorID - ID персонажа, необязательный.
Функция GetInvestmentGold возвращает количество инвестированного актером ActorID золота.
См. также: SetInvestmentGold
Относится к типу: Statistics Functions | Condition Functions

GetIsAlerted

Синтаксис:
CODE
[ActorID.]GetIsAlerted

Функция GetIsAlerted возвращает истину, если вызывающий актер (ActorID) находится в состоянии тревоги.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetIsClass

Синтаксис:
CODE
[ActorID.]GetIsClass ClassID

Пример:
CODE
GetIsClass Bard

Функция GetIsClass возвращает 1, если вызывающий актер (ActorID) относится к указанному в виде параметра классу (ClassID).
Относится к типу: Actor State Functions | Condition Functions

GetIsClassDefault

Синтаксис:
CODE
GetIsClassDefault ClassID

Пример:
CODE
GetIsClassDefault Acrobat

Функция GetIsClassDefault возвращает истину, если «класс по-умолчанию» игрока совпадает с указанным классом ClassID. «Класс по-умолчанию» — тот класс, который игра присваивает игроку на основании использования навыков в режиме «создания персонажа» (см. SetInCharGen).
Эти классы разбиваются следующим образом при использовании навыков Боя, Скрытности и Магии:
CODE
Класс / Бой / Магия / Скрытность
Thief / - / - / 7 /
Agent / - / 1 / 6 /
Assassin / 1 / 1 / 5 /
Acrobat / 2 / - / 5 /
Monk / 2 / 1 / 4 /
Pilgrim / 3 / - / 4 /
Bard / 2 / 2 / 3 /
Warrior / 7 / - / - /
Barbarian / 6 / - / 1 /
Crusader / 5 / 2 / - /
Knight / 5 / 1 / 1 /
Scout / 4 / 1 / 2 /
Archer / 4 / - / 3 /
Rogue / 3 / 2 / 2 /
Mage / - / 7 / - /
Sorcerer / 1 / 6 / - /
Healer / - / 5 / 2 /
Battlemage / 2 / 5 / - /
Witchhunter / 1 / 4 / 2 /
Spellsword / 3 / 4 / - /
Nightblade / 2 / 3 / 2 /

Класс по-умолчанию выбирается превалированием одного навыка над остальными. Поэтому, если один из них превысит 70%, вы станете воином (warrior), магом (mage) или вором (thief).
Например, если игрок набирает 45% боя, 30% магии, и 25% скрытности, то он становится Скаутом (Scout). Если появляется неопределенность с двумя вариантами, выигрывает больший. Например, при выборе между Варваром (Barbarian) и Крестоносцем (Crusader), выбор будет зависеть от % магии и скрытности.
Относится к типу: Player Functions | Condition Functions

GetIsCreature

Синтаксис:
CODE
[ActorID.]GetIsCreature

Функция GetIsCreature возвращает 1, если вызывающий актер (ActorID) — существо (creature).
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetIsCurrentPackage

Синтаксис:
CODE
[ActorID.]GetIsCurrentPackage PackageID

Примеры:
CODE
GetIsCurrentPackage AnvilGuardPatrolDay8x7

Функция GetIsCurrentPackage возвращает "1", если вызывающий актер (ActorID) в данный момент выполняет указанный в качестве параметра пакет AI (PackageID), и "0", если нет.
Примечания:
Это будет стопроцентно надежно лишь в том случае, когда у актера нет активных прерывающих пакетов.

Прерывающие пакеты, которые аннулируют текущий пакет с боем или диалогом (в т.ч., пакет раскрытия преступления), не влияют на результаты этой функции. Если актер выполняет пакет MyPackage и вступает в бой, то вызов функции "GetIsCurrentPackage MyPackage" возвратит истину (1).
"Запутанные" прерывающие пакеты, такие как пакет следования Follow Package (а также инициирующие диалог или преследование нарушителя), станут новым активным пакетом пока они не закончатся, поэтому GetIsCurrentPackage возвратит 0. Если вам нужно быть уверенным, что данный пакет не активен, вы должны проверить также процедуры искусственного интеллекта (AI Procedures).
Пример:
CODE
if getIsCurrentPackage MyPackage == 0
message "Пакет MyPackage не активный"
endif

Когда актер в настоящий момент следует за нарушителем границ или инициирует диалог, сообщение "Пакет MyPackage не активен" будет показано даже в том случае, если "MyPackage" является активным пакетом.
CODE
if getIsCurrentPackage MyPackage == 0
if getCurrentAIProcedure != 4 && getCurrentAIProcedure != 15
message "Пакет MyPackage не активный"
endif
endif

Такое же сообщение - "Пакет MyPackage не активный", будет отображаться и тогда, когда пакет не активный и актер не инициализировал диалог и не преследует преступника.
Относится к типу: AI Functions | Condition Functions | Actor Functions

GetIsCurrentWeather

Синтаксис:
CODE
GetIsCurrentWeather WeatherID
getweather WeatherID

Тип: вещественное число в диапазоне от 0 до 1
Возможные значения:
0 - текущая погода <> WeatherID
1 - текущая погода = WeatherID
другое - погода скоро сменится на WeatherID (даже если преобразование ещё не завершено)
Параметры: WeatherID - ID объекта погоды
Краткое описание: Сравнение текущего типа погоды и WeatherID
Пример:
CODE
GetIsCurrentWeather Thunderstorm

Функция GetIsCurrentWeather возвращает 1, если текущая погода соответствует заданной. Заметьте, что функция вернет истину, как только погода начнет меняться на заданную (даже если переход еще не завершен).
Относится к типу: Weather Functions | Condition Functions

GetIsGhost

Синтаксис:
CODE
[ActorID.]GetIsGhost

Функция GetIsGhost возвращает 1, если вызывающий актер (ActorID) находится в состоянии «призрака» ("ghost").
См. также: SetGhost
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetIsID

Синтаксис:
CODE
[RefID.]GetIsID ObjectID

Функция GetIsID возвращает 1, если вызывающий объект (RefID) - копия указанного явно базового объекта (ObjectID), как это определено в объектном окне конструктора (Object Window).
Пример:
CODE

```
Begin Onequip Player      ; один скрипт на весь набор брони
if GetIsID mycuirass == 1
message "Кираса надета "
elseif GetIsID myboots == 1
message "Ботинки надеты"
elseif GetIsID mygauntlets == 1
message "Наручи надеты"
elseif GetIsID mygreaves == 1
message "Поножи одеты"
elseif GetIsID myhelmet == 1
message "Шлем надет"
else
message "Щит надет"
endif
end
```

Относится к типу: Actor State Functions | Condition Functions

GetIsPlayableRace

Синтаксис:
CODE
[ActorID.]GetIsPlayableRace

Функция GetIsPlayableRace возвращает 1, если раса вызывающего актера ActorID играбельна.
См. также: Races.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetIsPlayerBirthsign

Синтаксис:
CODE
GetIsPlayerBirthsign BirthsignID
getPBS BirthsignID

Функция GetIsPlayerBirthsign возвращает 1, если знак рождения игрока совпадает с указанным знаком созвездия BirthsignID.
Относится к типу: Player Functions | Condition Functions

GetIsRace

Синтаксис:
CODE
[ActorID.]GetIsRace RaceID

Функция GetIsRace возвращает 1, если раса вызывающего актера (ActorID) совпадает с указанной явно расой RaceID.
Относится к типу: Actor State Functions | Condition Functions

GetIsReference

Синтаксис:
CODE
[ActorID|ObjectID.]GetIsReference ReferenceID

Пример:
CODE
```
if GetIsReference NPC1 == 1
messagebox "Я - NPC 1"
elseif GetIsReference NPC2 == 1
messagebox "Я - NPC 2"
else
messagebox "Я - NPC 3"
endif
```

Функция GetIsReference возвращает 1, если вызывающий объект (ObjectID) или актер (ActorID) являются копией указанного явно объекта (ReferenceID).
По сравнению с функцией GetIsID, которая сверяет с базовым объектом ID, определенным в окне Object Window конструктора, эта функция берет d в качестве параметра для сравнения указанную ref-переменную (копию объекта). Ссылка должна быть реальной.

Относится к типу: Object Functions | Condition Functions

GetIsSex

Синтаксис:
CODE
[ActorID.]GetIsSex Male|Female

Примеры:
CODE
If GetIsSex Male == 1
Message "Вы юноша, Жорж."
endif

If Player.GetIsSex Female
Message "Однако, вы очень находчивы."
endif

CODE
If Player.GetIsSex Female == 1
Message "О-о-о... да вы девушка!"
else
Message "Что это вы вырядились в дамские панталоны, молодой человек?"
endif

Функция GetIsSex возвращает 1, если пол вызывающего актера (ActorID) совпадает с указанным явно в виде параметра полом (Male - мужчина или Female - женщина).
См. также: GetPCIsSex
Относится к типу: Actor State Functions | Condition Functions

GetIsUsedItem

Синтаксис:
CODE
[ActorID.]GetIsUsedItem ObjectID

Пример:
CODE
GetIsUsedItem Apple01

Функция GetIsUsedItem возвращает истину, если вызывающий актер (ActorID) в данный момент использует указанный явно в виде параметра ObjectID предмет. Полезна ТОЛЬКО в функциях условия диалога, потому что переменная UsedItem всегда очищена при выполнении скрипта, но любое действие, вызываемое работой AI (выбор анимации, разговор, и т.д.) при установленной переменной, дает возможность проверить ее в условии.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetIsUsedItemType

Синтаксис:
CODE
[ActorID.]GetIsUsedItemType ObjectType

Пример:
CODE
GetIsUsedItemType Ingredient

Функция GetIsUsedItemType возвращает 1, если вызывающий актер (ActorID) в данный момент использует указанный явно в качестве параметра ObjectType тип предмета. Полезна ТОЛЬКО в функциях условия диалога, потому что переменная UsedItem всегда очищена при выполнении скрипта, но любое действие, вызываемое работой AI (выбор анимации, разговор, и т.д.) при установленной переменной дает возможность проверить ее в условии.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetItemCount

Синтаксис:
CODE
[ActorID|ContainerID.]GetItemCount ObjectID

Пример:
CODE
if (GetItemCount Gold001 <= 0)
disable
endif

CODE
Ref MyItem
Short count

set MyItem to ArenaAkaviriLongSword
if player.GetItemCount MyItem == 0
player.additem MyItem 1
endif

Функция GetItemCount возвращает количество указанных явно в виде параметра предметов ObjectID в инвентаре вызывающего актера (ActorID) или в контейнере (ContainerID).
Примечание:
• В качестве ObjectID можно использовать Ref-переменную.
См. также: AddItem, RemoveItem
Относится к типу: Object Functions | Condition Functions | Actor Functions

GetKnockedState

Синтаксис:
CODE
GetKnockedState ActorID

Функция GetKnockedState возвращает текущий статус нокаута указанного явно в виде параметра актера (ActorID):
CODE
0 = обычный
1 = нокаут
2 = нокдаун

Относится к типу: Actor State Functions | Condition Functions

GetLOS

Синтаксис:
CODE
[ActorID.]GetLOS ObjectID

Пример:
CODE
testActor.GetLOS TestMarker

Функция GetLOS возвращает 1, если вызывающий актер (ActorID) находится в на линии видимости указанного явно в виде параметра объекта (ObjectID). Может вызываться только на актерах.
Относится к типу: Movement Functions | Condition Functions | Actor Functions

GetLevel

Синтаксис:
CODE
[ActorID.]GetLevel

Функция GetLevel возвращает текущий уровень вызывающего актера (ActorID).
Относится к типу: Statistics Functions | Condition Functions | Actor Functions

GetLockLevel

Синтаксис:
CODE
[ObjectID.]GetLockLevel

Функция GetLockLevel возвращает уровень запирания вызывающего объекта (ObjectID) в диапазоне (0 ... 100). Заметьте, что функция не дает текущего состояния объекта, она показывает лишь тот уровень запирания, который был бы, если бы он был заперт.

См. также: GetLocked, Lock, Unlock
Относится к типу: Miscellaneous Functions | Condition Functions

GetLocked

Синтаксис:
CODE
[ObjectID.]GetLocked

Функция GetLocked возвращает 1, если вызывающий объект (ObjectID) заперт, и 0 в противном случае.
См. также: GetLockLevel, Lock, Unlock
Относится к типу: Miscellaneous Functions | Condition Functions

GetNoRumors

Синтаксис:
CODE
[ActorID.]GetNoRumors

Функция GetNoRumors возвращает 1, если у вызывающего актера (ActorID) флаг "No Rumors" установлен в "1" (т.е., у него в диалогах нет темы "Rumors" (слухи)).
Относится к типу: Actor State Functions | Condition Functions

GetOffersServicesNow

Синтаксис:
CODE
[ActorID.]GetOffersServicesNow

Функция GetOffersServicesNow возвращает 1, если вызывающий актер (ActorID) в текущем пакете предлагает услуги.
Относится к типу: Actor State Functions | Condition Functions

GetOpenState

Синтаксис:
CODE
[ObjectID.]GetOpenState

Функция GetOpenState возвращает статус открывания анимированной двери (ObjectID). Возвращаются следующие значения:
CODE
0 = ничего (не дверь)
1 = открыто
2 = открывается
3 = закрыто
4 = закрывается

См. также: SetOpenState
Относится к типу: Miscellaneous Functions | Condition Functions

GetPCExpelled

Синтаксис:
CODE
GetPCExpelled FactionID

Пример:
CODE
GetPCExpelled DarkBrotherhood

Функция GetPCExpelled возвращает 1, если игрок изгнан из указанной явно в виде параметра фракции FactionID.
См. также: SetPCExpelled
Относится к типу: Faction Functions | Player Functions | Condition Functions

GetPCFactionAttack

Синтаксис:
CODE
GetPCFactionAttack FactionID

Пример:
CODE
GetPCFactionAttack MagesGuild

Функция GetPCFactionAttack возвращает 1, если игрок атаковал члена указанной в виде параметра фракции (FactionID). Сбрасывается флаг функцией SetPCFactionAttack.
Относится к типу: Faction Functions | Player Functions | Condition Functions

GetPCFactionMurder

Синтаксис:
CODE
GetPCFactionMurder FactionID

Пример:
CODE
GetPCFactionMurder MagesGuild

Функция GetPCFactionMurder возвращает 1, если игрок убил члена указанной в виде параметра фракции (FactionID). Заметьте, что это не обязательно должно быть преступление, замеченное свидетелем. Сбрасывается флаг функцией SetPCFactionMurder.
Относится к типу: Faction Functions | Player Functions | Condition Functions

GetPCFactionSteal

Синтаксис:
CODE
GetPCFactionSteal FactionID

Пример:
CODE
GetPCFactionSteal MagesGuild

Функция GetPCFactionSteal возвращает 1, если игрок что-нибудь украл у члена указанной в виде параметра фракции (FactionID). Заметьте, что это не обязательно должно быть преступление, замеченное свидетелем. Сбрасывается флаг функцией SetPCFactionSteal.
Относится к типу: Faction Functions | Player Functions | Condition Functions

GetPCFactionSubmitAuthority

Синтаксис:
CODE
GetPCFactionSubmitAuthority GuardFactionID

Пример:
CODE
GetPCFactionSubmitAuthority ICFaction

Немного общей информации.
Внутренний флаг «faction submit authority» автоматически устанавливается в "1" всякий раз, когда персонаж игрока подчиняется (идет в тюрьму или платит штраф) страже из определенной фракции. (Этот флаг можно также установить самостоятельно, использовав функцию SetPCFactionSubmitAuthority. Для очистки (сброса флага) используйте эту функцию с параметром iFlag=0.)
Когда флаг устанавливается в "1" для конкретной фракции стражей, арестовавшей игрока, то ее можно можно в дальнейшем определить. В случае, если арестовавший страж принадлежит к нескольким фракциям, флаг устанавливается для первой по списку.
Описание функции GetPCFactionSubmitAuthority.
Функция GetPCFactionSubmitAuthority возвращает "1", если указанная явно в виде параметра фракция GuardFactionID совпадает с фракцией арестовавшего игрока стража. Таким образом можно определить, страж какой фракции арестовал игрока.
Относится к типу: Faction Functions | Player Functions | Condition Functions

GetPCFame

Синтаксис:
CODE
GetPCFame

Пример:
CODE
Set TempVar to GetPCFame

Функция GetPCFame возвращает текущее значение положительной репутации (fame) персонажа игрока. Только этот персонаж имеет репутацию - как положительную, так и отрицательную.
Относится к типу: Player Functions | Condition Functions

GetPCInFaction

Синтаксис:
CODE
GetPCInFaction

То же, что и функция GetInFaction, но применима только к игроку. Основное использование — функция состояния в диалогах.
Относится к типу: Player Functions | Condition Functions

GetPCInfamy

Синтаксис:
CODE
GetPCInfamy

Пример:
CODE
Set TempVar to GetPCInfamy

Функция GetPCInfamy возвращает текущее значение отрицательной репутации (infamy) игрока. Только у игрока может быть как положительная, так и отрицательная репутация.
Относится к типу: Player Functions | Condition Functions

GetPCIsClass

Синтаксис:
CODE
GetPCIsClass

То же, что и GetIsClass, но применима только к игроку. Основное использование — функция состояния в диалогах.
Относится к типу: Player Functions | Condition Functions

GetPCIsRace

Синтаксис:
CODE
GetPCIsRace

То же, что и GetIsRace, но применима только к игроку. Основное использование — функция состояния в диалогах.
Относится к типу: Player Functions | Condition Functions

GetPCIsSex

Синтаксис:
CODE
GetPCIsSex Male|Female

Функция GetPCIsSex возвращает 1, если персонаж игрока относится к указанному явно в виде параметра полу - Male(мужчина) или Female (женщина).
(В основном используется как условие в функциях диалогов, где игрок не является «целью»).
Относится к типу: Player Functions | Condition Functions

GetPCMiscStat

Синтаксис:
CODE
GetPCMiscStat MiscStatID

Пример:
CODE
GetPCMiscStat 4; возвращает самое большое пожертвование игрока

Функция GetPCMiscStat возвращает значение указанной в виде параметра прочей статистики игрока (MiscStatID).
ID прочей статистики:
0 HOURS IN PRISON - время, проведенное в тюрьме
1 DAYS PASSED - пройденных дней
2 SKILL INCREASES - рост мастерства
3 TRAINING SESSIONS - тренировочные сессии
4 LARGEST BOUNTY - самые большие пожертвования
5 CREATURES KILLED - количество убитых существ
6 PEOPLE KILLED - количество убитых людей
7 PLACES DISCOVERED - обнаруженных мест
8 LOCKS PICKED - выбрано замков
9 PICKS BROKEN - разбито указателей
10 SOULS TRAPPED - поймано душ
11 INGREDIENTS EATEN - съеденные ингредиенты
12 POTIONS MADE - изготовлено текстур
13 OBLIVION GATES SHUT - закрыто ворот Обливиона
14 HORSES OWNED - владение лошадьми
15 HOUSES OWNED - владение домами
16 STORES INVESTED IN - сохранность инвестиций
17 BOOKS READ - прочитано книг
18 SKILL BOOKS READ - прочитано книг, увеличивающих навыки
19 ARTIFACTS FOUND - найдено артефактов
20 HOURS SLEPT - количество часов, проведенных во сне
21 HOURS WAITED - количество часов, проведенных в ожидании
22 DAYS AS A VAMPIRE - количество дней, проведенных "в шкуре" вампира
23 LAST DAY AS VAMPIRE - последний лень, проведенный как вампир
24 PEOPLE FED ON - накормленных людей
25 JOKES TOLD - рассказано шуток
26 DISEASES CONTRACTED - вылечено болезней
27 NIRNROOTS FOUND - найдено нирнрутов
28 ITEMS STOLEN - украдено предметов
29 ITEMS PICKPOCKETED - присвоено найденных предметов
30 TRESPASSES - количество нарушений границ
31 ASSAULTS - количество нападений
32 MURDERS - количество убийств
33 HORSES STOLEN - украдено лошадей

Примечание: Заметьте, что при вводе в качестве параметра MiscStatID чисел, больших чем 33, функция все равно возвращает число (по крайней мере, в консоли). Это ничего не значащие большие целые числа, так как, похоже, функция GetPCMiscStat n просто читает n-ый 4-байтный кластер после определенной точки и показывает его значение, как неподписанное длинное целое (unsigned long int).
Относится к типу: Player Functions | Condition Functions

GetPCSleepHours

Синтаксис:
CODE
GetPCSleepHours

Функция GetPCSleepHours возвращает число часов, которые игрок выставил в меню сна. Возвращает 0, если игрок не спит.
Относится к типу: Player Functions | Condition Functions

GetPackageTarget

Синтаксис:
CODE
set refVar to [ActorID.]GetPackageTarget

Функция GetPackageTarget возвращает ref-переменную цели текущего пакета вызывающего актера ActorID (если цель есть).
Относится к типу: Reference Variable Functions

GetParentRef

Синтаксис:

CODE
set refVar to [ObjectID.]GetParentRef

Функция GetParentRef возвращает указатель (ref-переменную) на родителя (базовый объект) вызывающего объекта ObjectID. Если такового нет, возвращается "0"
Относится к типу: Reference Variable Functions

GetPersuasionNumber

К сожалению, описания функции GetPersuasionNumber (а также GetTotalPersuasionNumber и некоторых других) в официальном WIKI нет. Ничего не дал и поиск в интернете - ни на английских сайтах, ни на любых других.
Функции GetPersuasionNumber и GetTotalPersuasionNumber в скриптах Обливиона не используются, увы...
В WIKI на страницах с этими функциями приведено только слово Placeholder, которое можно перевести как зарезервированное место, т.е., если вам удастся их расшифровать, вы вполне можете заполнить пустующие страницы.
Если Вы желаете "расшифровать" эти загадочные функции, прочитайте эту тему.
См. также: GetTotalPersuasionNumber
Относится к типу: Miscellaneous Functions | Condition Functions

GetPlayerControlsDisabled

Синтаксис:
CODE
GetPlayerControlsDisabled

Функция GetPlayerControlsDisabled возвращает 1, если управление персонажа игрока отключено.
См. также: DisablePlayerControls, EnablePlayerControls
Относится к типу: Player Functions | Condition Functions

GetPlayerHasLastRiddenHorse

Синтаксис:
CODE
GetPlayerHasLastRiddenHorse

Функция GetPlayerHasLastRiddenHorse возвращает 1, если у игрока есть «последняя лошадь, на которой ездили» — т.е. у игрока есть лошадь, он на ней ездил и она все еще жива.
См. также: IsPlayersLastRiddenHorse
Относится к типу: Player Functions | Condition Functions

GetPos

Синтаксис:
CODE
[ObjectID.]GetPos axis

Пример:
CODE
GetPos Z

Функция GetPos возвращает текущие мировые координаты объекта (ObjectID) на определенной оси (X, Y, или Z) в виде вещественного числа (float).
Относится к типу: Movement Functions | Condition Functions

GetQuestRunning

Синтаксис:
CODE
GetQuestRunning QuestID
GetQR QuestID

Пример:
CODE
GetQuestRunning MS29

Функция GetQuestRunning возвращает "1", если указанный в виде параметра квест (QuestID) в данный момент выполняется, и "0" в противном случае.
См. также: StartQuest, StopQuest
Относится к типу: Quest Functions | Condition Functions

GetQuestVariable

Синтаксис:
CODE
GetQuestVariable

Функция GetQuestVariable возвращает значение квестовой переменной.
Примечание: Это ТОЛЬКО функция УСЛОВИЯ. Чтобы получить значение квестовой переменной в скрипте, нужды в этой функции нет. Просто используйте следующий синтаксис:
CODE
QuestName.VarName

См. также: GetGlobalValue, GetScriptVariable, GetStage
Относится к типу: Quest Functions | Condition Functions

GetRandomPercent

Синтаксис:
CODE
GetRandomPercent

Функция GetRandomPercent возвращает случайное число в интервале от 0 до 99 включительно.
Чтобы сгенерировать случайное число в заданном диапазоне между мин. и макс. значениями используйте следующий синтаксис:
CODE
set randVal to min + GetRandomPercent * (max-min) / 99

Примеры:
CODE
short dice
set dice to 1 + 0.06 * GetRandompercent; => 1 to 6 (almost equal chances see talk page)

CODE
short rnd
set rnd to 5.0/99 * Getrandompercent ; => 0 to 5 (only 1% chance for 5! see talk page)

CODE
float rnd
set rnd to 0.05 * Getrandompercent ; => 0.00 to 4.95

CODE
float rnd
set rnd to 5/99 * Getrandompercent ; => 0.00 to 5.00

См. также: Rand(OBSE)
Относится к типу: Miscellaneous Functions | Condition Functions

GetRestrained

Синтаксис:
CODE
[ActorID.]GetRestrained

Функция GetRestrained возвращает "1", если вызывающий актер (ActorID) находится в бессознательном состоянии (restrained state).
Смотрите также: SetRestrained
Относится к типу: AI Functions | Condition Functions

GetScale

Синтаксис:
CODE
[ObjectID.]GetScale

Тип возвращаемой переменной: вещественное, неотрицательное.
Описание: Функция GetScale возвращает масштабный множитель вызывающего объекта (ObjectID), на который умножен его нормальный размер.

Возможные значения:
меньше 1.0 (0.5 ... 0.99) - уменьшенные размеры (для актеров - рост)
1.0 - нормальные, заданные изначально, размеры.
больше 1.0 (1.01 ... 2.0) - увеличенные размеры
Примечание: Стандартный рост актера в империи равен 128 ед. (units).
Относится к типу: Statistics Functions | Condition Functions

GetScriptVariable

Синтаксис:
CODE
GetScriptVariable ObjectRef Variable

Функция GetScriptVariable возвращает значение локальной переменной Variable из скрипта на указанном в виде параметра объекте ObjectRef.
Примечание: Это ТОЛЬКО функция УСЛОВИЙ! Чтобы получить значение объектной переменной в скрипте, нужды в этой функции нет. Просто используйте следующий синтаксис:
CODE
ObjectRef.VarName

См. также: GetGlobalValue, GetQuestVariable
Относится к типу: Miscellaneous Functions | Condition Functions

GetSecondsPassed

Синтаксис:
CODE
GetSecondsPassed

Тип возвращаемого значения: Float (вещественное)
Параметры: нет
Примеры:
CODE
GetSecondsPassed

Функция GetSecondsPassed возвращает число секунд, прошедших с момента последнего вызова этой функции в предыдущих игровых фреймах. Тип возвращаемой переменной - вещественный (float).
Очень полезна для создания таймера внутри скрипта. У каждого скрипта есть свой счетчик для GetSecondsPassed и при вызове этой функции он всегда сбрасывается в "0". Таким образом, если ее вызывать несколько раз подряд внутри одного фрейма, то эта функция в скрипте будет возвращать "0".
Пример 1:
CODE
float timer
begin gamemode
if timer > 0
set timer to timer - GetSecondsPassed
else
; Время истекло! Делаем что-то.
endif
end

Пример 2:
CODE
float timer1
float timer2
begin gamemode
set timer1 to getsecondspassed
; Здесь возвращается время, прошедшее
; с момента вызова функции в предыдущем фрейме
set timer2 to getsecondspassed; Здесь возвращается 0
end

Относится к типу: Time Functions

GetSelf

Синтаксис:
CODE

set refVar to GetSelf
set refVar to this

Пример:
CODE
if GetSelf == GetActionRef

CODE
set MyQuest.targetRef to GetSelf

Функция GetSelf возвращает ref-переменную в качестве указателя на вызывающий объект. Используется для утверждений или установки ref-переменных в других скриптах.
Примечания:
Когда эта функция вызывается на объектах, которые можно положить в контейнер (например, оружие или предметы), ref-переменная будет возвращаться, но лишь до тех пор, пока объект остается в игровом мире.
Эта функция работает ненадежно, когда вызывается на игроке.
Относится к типу: Reference Variable Functions

GetShouldAttack

Синтаксис:
CODE
[ActorID.]GetShouldAttack TargetActor

Пример:
CODE
EvilBanditBoss.GetShouldAttack player

Функция GetShouldAttack возвращает число, соответствующее желанию вызывающего актера (ActorID) атаковать указанную в виде параметра цель TargetActor. Если возвращаемое значение больше нуля, то актер будет атаковать цель при обычных обстоятельствах (т.е., актер агрессивен, может обнаружить цель, не задержан, не находится в бессознательном состоянии и т.п.).
Относится к типу: Combat Functions | Condition Functions

GetSitting

Синтаксис:
CODE
[ActorID.]GetSitting

Функция GetSitting возвращает для вызывающего актера (ActorID) в зависимости от его состояния по отношению к действию "сесть/встать" следующие значения :
0 – Не сидит
1 – Загружается анимация "сидения"
2 – Готов сесть
3 - Сидит
4 – Готов встать
11 – Загружается анимация "езды на лошади"
12 – Готов сесть в седло
13 – Сидит на лошади
14 – Готов слезть с лошади

Примечание: Эта функция не будет возвращать "1" или "11", если вызывается на персонаже игрока.
Относится к типу: Actor State Functions | Condition Functions

GetSleeping

Синтаксис:
CODE
[ActorID.]GetSleeping

Функция GetSleeping возвращает следующие базовые значения в зависимости от состояния сна вызывающего актера (ActorID):
0 – Не спит
1 – Загружается пакет сна
2 – Готов спать
3 - Спит
4 – Готов проснуться

Примечание: Функция не работает на игроке, так как основана на состояния анимации актера. Используйте для этого функцию IsPCSleeping.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

GetStage

Синтаксис:
CODE
GetStage QuestID

Пример:
CODE
GetStage FG01

Функция GetStage возвращает текущее максимальное состояние завершенности указанного в виде параметра квеста QuestID. Например, если закончены стадии 10, 30 и 75, функция GetStage вернет 75. Это будет справедливо и в том случае, если стадия 30 была выполнена уже после завершения стадии 75 - все равно будет возвращаться значение 75 (т.е., максимальное). Если же вам нужно знать, какая конкретно стадия квеста завершена последней, используйте функцию GetStageDone.
См. также: SetStage, GetStageDone
Относится к типу: Quest Functions | Condition Functions

GetStageDone

Синтаксис:
CODE
GetStageDone QuestID StageIndex

Пример:
CODE
GetStage MS09 30

Функция GetStageDone возвращает "1", если указанный этап StageIndex квеста QuestID завершен, и "0" в противном случае.
См. также: GetStage, SetStage
Относится к типу: Quest Functions | Condition Functions

GetStartingAngle

Синтаксис:
CODE
[ObjectID.]GetStartingAngle axis

Примеры:
CODE
GetStartingAngle Z

Функция GetStartingAngle возвращает первоначальный угол поворота объекта ObjectID по выбранной оси (X, Y, или Z) относительно мировых координат. Тип возвращаемой переменной - вещественный (float). Угол возвращается тот, который был изначально задан в редакторе, а не текущий, который мог измениться в игровом процессе.
Относится к типу: Movement Functions | Condition Functions

GetStartingPos

Синтаксис:
CODE
[ObjectID.]GetStartingPos axis

Пример:
CODE
GetStartingPos Z

Функция GetStartingPos возвращает первоначальное местоположение (координату) вызывающего объекта ObjectID по выбранной оси (X, Y, или Z) относительно мировых координат. Тип возвращаемой переменной - вещественное (float). Координата возвращается та, которая была задана в редакторе изначально, в конструкторе, а не текущая, которая могла в игровом процессе измениться.
Относится к типу: Movement Functions | Condition Functions

GetTalkedToPC

Синтаксис:
CODE
[ActorID.]GetTalkedToPC

Функция GetTalkedToPC возвращает "1", если вызывающий актер (ActorID) уже разговаривал с персонажем игрока, и "0" в противном случае.
Относится к типу: Actor State Functions | Condition Functions

GetTalkedToPCParam

Синтаксис:
CODE
GetTalkedToPCParam ActorID

Пример:
CODE
GetTalkedToPCParam ValenDrethRef

Функция GetTalkedToPCParam возвращает "1", если указанный в виде параметра актер (ActorID) уже говорил с персонажем игрока, и "0" в противном случае.
Используется, в основном, как функция условия в диалогах.
GetTalkedToPCParam относится к типу Non-Reference Functions, которые принимают актера в качестве параметра как ref-переменную, вместо выполнения на актере.
См. также: GetTalkedToPC
Относится к типу: Actor State Functions | Condition Functions

GetTimeDead

Синтаксис:
CODE
[ActorID.]GetTimeDead

Пример:
CODE
BossBanditRef.GetTimeDead

Функция GetTimeDead возвращает время (в часах) с момента смерти вызывающего актера ActorID. Например, если BossBanditRef мертв уже 3,5 часа, то вызов BossBanditRef.GetTimeDead вернет 3,5.
Относится к типу: Actor State Functions | Condition Functions

GetTotalPersuasionNumber

К сожалению, на официальном ВИКИ (да и в интернете) отсутствует описание этой функции, как и сходной - GetPersuasionNumber... Более подробно об этом смотрите в ее описании.
А в Вики значится Placeholder, что можно перевести как "зарезервировано".
См. также: GetPersuasionNumber
Относится к типу: Miscellaneous Functions | Condition Functions

GetTrespassWarningLevel

Синтаксис:
CODE
GetTrespassWarningLevel

Функция GetTrespassWarningLevel возвращает текущий уровень предупреждений о нарушении частной собственности.
Обычно используется совместно с функцией IsTrespassing.
Уровни предупреждений о нарушении частной собственности:
0: Первое предупреждение
1: Второе предупреждение
2: Сообщено о нарушении, стража уже в пути
Относится к типу: Crime Functions | Condition Functions

GetUnconscious

Синтаксис:
CODE
[ActorID.]GetUnconscious

Функция GetUnconscious возвращает "1", если актер находится в бессознательном состоянии, и "0" в противном случае.
См. также: SetUnconscious
Относится к типу: AI Functions | Condition Functions

GetUsedItemActivate

Синтаксис:
CODE
GetUsedItemActivate

К сожалению, в официальном Вики нет описания этой функции. Значится только "Placeholder" - зарезервировано.
Относится к типу: Condition Functions

GetUsedItemLevel

Синтаксис:
CODE
[ItemID.]GetUsedItemLevel

Функция GetUsedItemLevel возвращает "0", если вызывающий предмет ItemID находится на уровне земли, и "1", если на уровне пояса и выше. Используется исключительно как функция условия в диалоговом окне Idle manager конструктора для определения необходимой анимации.
Относится к типу: Condition Functions

GetVampire

Синтаксис:
CODE
[NpcID.]GetVampire

Функция GetVampire возвращает "1", если вызывающий актер (NpcID) — вампир, и "0" в противном случае.
См. также: VampireFeed, HasVampireFed
Относится к типу: Actor State Functions | Condition Functions

GetWalkSpeed

Синтаксис:
CODE
[ActorID.]GetWalkSpeed

Параметры: ActorID - ID персонажа, необязательный.
Функция GetWalkSpeed позволяет узнать скорость ходьбы вызывающего актера (ActorID).
Относится к типу: Statistics Functions | Condition Functions

GetWeaponAnimType

Синтаксис:
CODE
GetWeaponAnimType ActorID

Функция GetWeaponAnimType возвращает тип анимации текущего оружия вызывающего актера (ActorID), указанного явно в виде параметра.
Типы анимаций:
0 = рукопашная (нет оружия)
1 = одноручное оружие
2 = двуручное оружие
3 = лук
Относится к типу: Actor State Functions | Condition Functions

GetWeaponSkillType

Синтаксис:
CODE
[ActorID.]GetWeaponSkillType TypeWeapont

Примеры:
CODE

GetWeaponSkillType 1; Возвращение текущего навыка использования клинков

Функция GetWeaponSkillType возвращает для вызывающего актера (ActorID) его текущий навык в использовании указанного в виде параметра (TypeWeapont) типа оружия:
CODE
0 = Нет оружия (рукопашная - Hand-to-Hand) или нет навыка (персонал - Staff)
1 = Клинки (Blade)
2 = Дробящее оружие (Blunt)
3 = Меткость стрельбы лука (Marksman)

Пример 2.
CODE
short charUsingBow

Begin OnPackageChange (Or other one-time condition like OnStartCombat)
if GetWeaponSkillType 3;(Checks if actor is using marksman skill)
Set charUsingBow to 1
else
Set charUsingBow to 0
endif
End
Полный текст скрипта и его использование см. на официальном WIKI в разделе Questions в статье "Increasing Bow Damage"
Относится к типу: Actor State Functions | Condition Functions

GetWindSpeed

Синтаксис:
CODE
GetWindSpeed

Функция GetWindSpeed возвращает скорость ветра при текущей погоде (в пределах от 0.0 до 1.0, как установлено в редакторе в настройках объекта погоды (Weather type)). Тип возвращаемой переменной - вещественное.
Относится к типу: Weather Functions | Condition Functions

GoToJail

Синтаксис:
CODE
GoToJail

Функция GoToJail взимает штраф (crime gold), конфискует все украденные предметы из инвентаря игрока и отправляет в ближайшую тюрьму.
Примечания:
Функция работает только на персонаже игрока.
После того, как игрок будет перемещен в тюрьму на новой локации игра будет ожидать до тех пор, пока игрок не отдохнет, и только затем телепортирует его к выходу из тюрьмы.
Тюрьма содержит пару связанных дверей-маркеров ("Prisonmarker"). Один маркер установлен непосредственно в интерьере тюремной камеры, а второй - во внешней локации, куда персонаж игрока будет перемещен после отбывания наказания.
Конфискованные у игрока ворованные предметы помещены в ящик "Ворованные предметы" (Stolengoods), который находится в той же внутренней ячейке тюрьмы, что и маркер (Prisonmarker).
См. также: PayFine
Относится к типу: Crime Functions

H

HasFlames

Синтаксис:
CODE
[ObjectID.]HasFlames

Функция HasFlames возвращает "1", если к вызываемому объекту (ObjectID) уже добавлены объекты пламени FlameNode.
О FlameNode можно читайте здесь: http://cs.elderscrolls.com/constwiki/index.php/FlameNode
См. также: FlameNode, CanHaveFlames, AddFlames, RemoveFlames
Относится к типу: Object Functions | Condition Functions

HasMagicEffect

Синтаксис:
CODE
[ObjectID.]HasMagicEffect EffectID

Пример:
CODE
HasMagicEffect FIDG

Функция HasMagicEffect возвращает "1", если вызывающий объект (ObjectID) в данный момент подвержен воздействию указанного в виде параметра (EffectID) магического эффекта.
См. также: IsSpellTarget, OnMagicEffectHit, Magic Effects List
Относится к типу: Magic Functions | Condition Functions | Actor Functions

HasVampireFed

Синтаксис:
CODE
[ActorID|Player.]HasVampireFed

Функция HasVampireFed возвращает "1", если у вызывающего персонажа, на котором вызвана функция, есть флаг "vampire fed" и он установлен в истину (в "1", его устанавливает игра по окончании пакета "vampire feed").
Примечания:
Функция работает корректно только на игроке, т.к. данный флаг есть только в его свойствах. Поэтому при вызове функции ActorID.HasVampireFed будет всегда возвращаться "0", для любых актеров.
HasVampireFed также сбрасывает флаг, поэтому, вызвав ее два раза подряд, в первый раз она вернет "1", а во второй раз — "0".
См. также: VampireFeed
Относится к типу: Player Functions

I

IsActionRef

Синтаксис:
CODE
IsActionRef ObjectRefID

Пример:
CODE
IsActionRef player

Функция IsActionRef используется только внутри блока OnActivate. Возвращает "1", если указанный в виде параметра объект (ObjectRefID) активировал что-то или кого-то (т.е., был активатором).
Пример скрипта:
CODE
begin OnActivate
if IsActionRef player == 1
MessageBox "Ты не можешь активировать меня. Слабак!"
else
Activate
endif
end

Вот пример реального скрипта, который приводится в туториале "A beginner's guide, lesson 5 - Anatomy of a quest, part 2" в стадии квеста 90:
CODE
Scriptname MS02SecretDoor

Begin OnActivate
; блок стартует только после активизации двери.
; активизация возможна только после 88 стадии и записи в журнале

if (GetStage MS02 >= 110) && (IsActionRef Player == 1) Activate
; Эти условия означают, что дверь откроется только для Вас, как только вы завершите квест.
elseif (GetStage MS02 >= 90) && (IsActionRef VelwynBenirusRef == 1) Activate
; Этот условие означает, что дверь откроется только для Велвина, пока вы не завершите свой квест
endif
End

Begin GameMode
Unlock
End

Относится к типу: Object Functions

IsActor

Синтаксис:
CODE
[ObjectID.]IsActor

Функция IsActor возвращает 1, если вызывающий объект (ObjectID) — актер (существо или NPC).
См. также: GetIsCreature
Относится к типу: Actor State Functions | Condition Functions

IsActorAVictim

Синтаксис:
CODE
[NpcID.]IsActorAVictim

Функция IsActorAVictim возвращает "1", если вызывающий NPC (NpcID) в данный момент является жертвой преступления (функция возвращает значение только для одного NPC за один раз). Используется, в основном, как условие для диалогов, связанных с преступлениями.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsActorDetected

Синтаксис:
CODE
[ActorID.]IsActorDetected

Функция IsActorDetected возвращает "1", если вызывающий актер (ActorID) замечен другим актером.
См. также: GetDetected, GetDetectionLevel, Category: Detection
Относится к типу: Crime Functions | Condition Functions | Actor Functions

IsActorEvil

Синтаксис:
CODE
[ActorID.]IsActorEvil

Функция IsActorEvil возвращает "1", если вызывающий актер (ActorID) принадлежит только к злым фракциям (т.е., по крайней мере хотя бы к одной злой фракции и ни к одной не злой).
Эта функция только работает с NPC, не с существами. Никакие существа не считаются злыми, независимо от того, к какой фракции они принадлежат.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsActorUsingATorch

Синтаксис:
CODE
[ActorID.]IsActorUsingATorch

Функция IsActorUsingATorch возвращает "1", если вызывающий актер (ActorID) использует факел (т.е., вытащил его).
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsActorsAIOff

Синтаксис:
CODE
[ActorID.]IsActorsAIOff

Функция IsActorsAIOff возвращает "1", если AI вызывающего актера (ActorID) был выключен командой ToggleActorsAI.
См. также: ToggleActorsAI
Относится к типу: Console Functions | Actor Functions

IsAnimPlaying

Синтаксис:
CODE
[ObjectID.]IsAnimPlaying

Функция IsAnimPlaying возвращает "1", если объект вызова (ObjectID) в текущий момент времени проигрывает анимацию.
Относится к типу: Animation Functions

IsCellOwner

Синтаксис:
CODE
IsCellOwner CellID NpcID/FactionID (optional)

Пример:
CODE
IsCellOwner WeynonPrioryHouse BladesFaction
IsCellOwner MyImperialCityHouse

Функция IsCellOwner возвращает "1", если указанной в виде параметра CellID ячейкой владеет NPC (NpcID) или фракция (FactionID). Если второй параметр не указывать, функция вернет "1", если ячейкой владеет игрок.
См. также: IsInMyOwnedCell
Относится к типу: Crime Functions | Condition Functions

IsCloudy

Синтаксис:
CODE
IsCloudy

Тип: вещественное в диапазоне от 0 до 1.
Функция IsCloudy возвращает "0", если текущую погоду нельзя классифицировать как облачную. Значение от 0 до 1 означает переход погоды к облачности. "1" означает установившуюся облачную погоду.
Относится к типу: Weather Functions | Condition Functions

IsContinuingPackagePCNear

Синтаксис:
CODE
[ActorID.]IsContinuingPackagePCNear

Функция IsContinuingPackagePCNear возвращает "1", если вызывающий актер на момент вызова является защищенным от переключения пакетов в связи с тем, что в текущем пакете отмечен флаг "Продолжать, если рядом PC" ("Continue if PC near"), т.е. актер хочет переключиться на новый пакет, но не может, т.к. игрок находится в этой же ячейке.
Относится к типу: AI Functions | Condition Functions | Actor Functions

IsCurrentFurnitureObj

Синтаксис:
CODE
[ActorID.]IsCurrentFurnitureObj FurnitureID

Пример:
CODE
Bob.IsCurrentFurnitureObj KingsThroneObject

Функция IsCurrentFurnitureObj возвращает "1", если NPC (ActorID) сидит на предмете мебели указанного в виде параметра типа (FurnitureID).
См. также: IsCurrentFurnitureRef
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsCurrentFurnitureRef

Синтаксис:
CODE
IsCurrentFurnitureRef RefID

Пример:
CODE
Bob.IsCurrentFurnitureRef TavernCornerChairRef

Функция IsCurrentFurnitureRef возвращает "1", если NPC сидит на указанном в виде параметра RefID предмете мебели.
См. также: IsCurrentFurnitureObj
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsEssential

Синтаксис:
CODE
[ActorID.]IsEssential

Функция IsEssential возвращает "1", если у вызывающего актера (ActorID) флаг "Essential" установлен в "1".
Напомним, что когда этот флаг установлен в "1", NPC/Creature убить нельзя, они могут лишь временно потерять сознание, когда их здоровье упадает до 0 и ниже. Таким образом персонаж помечается как "важная" для игры персона и даст ему защиту в том случае, если его случайно убьет игрок. Такая необходимость может возникнуть, например, в квестах для их нормального завершения.
См. также: SetEssential
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsFacingUp

Синтаксис:
CODE
[ActorID|Player.]IsFacingUp

Функция IsFacingUp используется в диалоговом окне менеджера анимации конструктора (Idle manager) в качестве условия и возвращает "1", если существо упало, находится в бессознательном состоянии и обращено "лицом" вверх.
Функция IsFacingUp возвращает "1", если вызывающий актер ActorID или персонаж игрока оседлал четвероногое животное, например, игрок на лошади.
Но если вы хотите использовать функцию на самом четвероногом существе, то используйте в таком случае функцию IsLeftUp.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsGuard

Синтаксис:
CODE
[NpcID.]IsGuard

Функция IsGuard возвращает "1", если класс вызывающего NPC (NpcID) отмечен как "Guard" (класс стражи).
См. также: Classes
Относится к типу: Crime Functions | Actor State Functions | Condition Functions | Actor Functions

IsHorseStolen

Синтаксис:
CODE
[HorseID.]IsHorseStolen

Функция IsHorseStolen возвращает "1", если вызывающий объект (HorseID) — украденная лошадь.
Относится к типу: Crime Functions | Condition Functions | Actor Functions

IsIdlePlaying

Синтаксис:
CODE
[ActorID.]IsIdlePlaying

Пример:
CODE
if (IsIdlePlaying==1 && MyQuest.HasTeleported==1);Idle animation bug
set MyQuest.HasTeleported to 0
EvaluatePackage
endif

Функция IsIdlePlaying возвращает "1", если вызывающий актер (ActorID) в данный момент проигрывает определенный idle.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsInCombat

Синтаксис:
CODE
[ActorID.]IsInCombat

Функция IsInCombat возвращает "1", если вызывающий актер (ActorID) находится в состоянии боя.
Относится к типу: Actor State Functions | Combat Functions | Condition Functions | Actor Functions

IsInDangerousWater

Синтаксис:
CODE
[ActorID.]IsInDangerousWater

Функция IsInDangerousWater возвращает "1", если вызывающий актер находится в опасной "воде" (например, в лаве).
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsInInterior

Синтаксис:
CODE
[ActorID.]IsInInterior

Функция IsInInterior возвращает "1", если вызывающий актер (ActorID) находится во внутренней ячейке (интерьере).
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsInMyOwnedCell

Синтаксис:
CODE
[ActorID.]IsInMyOwnedCell

Пример использования функции как условия:
CODE
If player.IsInMyOwnedCell == 1 && mycontainer.GetInSameCell player == 1 .....

Функция IsInMyOwnedCell возвращает "1", если вызывающий актер (ActorID) находится в ячейке, принадлежащей ему.
См. также: IsCellOwner
Относится к типу: Crime Functions | Condition Functions | Actor Functions

IsLeftUp

Синтаксис:
CODE
[CreatureID.]IsLeftUp

Пример:
CODE
player.IsInMyOwnedCell == 1 && mycontainer.GetInSameCell player == 1

Функция IsLeftUp используется как условие в диалоговом окне конструктора Idle manager и возвращает "1", если вызывающее четвероногое животное (CreatureID) находится в нокдауне и его левая сторона обращена вверх (лежит на правом боку).
Для двуногих используется функция IsFacingUp.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsOwner

Синтаксис:
CODE
[ObjectID.]IsOwner NPC/FactionID (optional)

Пример:
CODE

SpecialObject.IsOwner BladesFaction
SpecialObject.IsOwner AmuseiRef
SpecialObject.IsOwner

Функция IsOwner возвращает "1", если вызывающий объект (ObjectID) находится во владении указанной фракции/NPC (NPC/FactionID). Проверяется только истинное владение объектом, не принимая во внимание владельца ячейки. Если функция вызвана без параметра, то возвращается "1", если вызывающим объектом владеет игрок (т.е., владельца нет).
Этот код не работает:
CODE
IsOwner Player

Относится к типу: Object Functions | Condition Functions

IsPCAMurderer

Синтаксис:
CODE
[ActorID.]IsPCAMurderer

Функция IsPCAMurderer возвращает "1", если вызывающий актер (ActorID) когда-нибудь убивал NPC.
Относится к типу: Player Functions | Condition Functions

IsPCSleeping

Синтаксис:
CODE
IsPCSleeping

Функция IsPCSleeping возвращает "1", если персонаж игрока в данный момент спит.
Относится к типу: Player Functions | Condition Functions

IsPlayerInJail

Синтаксис:
CODE
IsPlayerInJail

Функция IsPlayerInJail возвращает "1", если персонаж игрока в данный момент находится «в тюрьме».
Относится к типу: Crime Functions | Player Functions | Condition Functions

IsPlayerMovingIntoNewSpace

Синтаксис:
CODE
IsPlayerMovingIntoNewSpace

Функция IsPlayerMovingIntoNewSpace возвращает "1", если персонаж игрока в данный момент находится в процессе перехода в новое пространство, т.е. использует fast travel или идет через загружающую дверь.
Применение этой функции может вызывать недоумение, но ее полезно помещать на любые пакеты follow, которые обусловлены местоположением игрока. Скажем, есть NPC, чей пакет follow за игроком определен так, чтобы он не заходил во внутренние ячейки и не покидал Skingrad. Проблема в том, что последователи перемещаются до того, как прибывает игрок, так что они не пересчитывают, что цель уже во внутренней ячейке или за пределами Skingrad, пока они и игрок не будут уже в новом месте — получаем NPC, который перемещается с вами, а затем топает пешком до Skingrad, где бы вы ни оказались.
Эта функция возвращает истину, когда игрок находится в середине загрузки в новую ячейку или в процессе fast travelling — так что, если поместить
CODE
IsPlayerMovingIntoNewSpace == 0
как условие для пакета follow, он пересчитается во время загрузки и не пойдет за игроком за Skingrad или во внутренние ячейки.
Относится к типу: Player Functions | Condition Functions

IsPlayersLastRiddenHorse

Синтаксис:
CODE
[ActorID.]IsPlayersLastRiddenHorse

Функция IsPlayersLastRiddenHorseвозвращает "1", если вызывающий актер (ActorID) — последняя лошадь, на которой ездил игрок.
См. также: GetPlayerHasLastRiddenHorse
Относится к типу: Player Functions | Condition Functions

IsPleasant

Синтаксис:
CODE
IsPleasant

Функция IsPleasant возвращает "0", если текущую погоду нельзя охарактеризовать как хорошую (солнечную, приятную).
Возвращаемое значение - вещественное в диапазоне от "0" до "1".
Промежуточное значение между "0" и "1" означает процесс улучшения погоды в сторону хорошей.
В случае хорошей и стабильной погоды будет возвращена "1".
Относится к типу: Weather Functions | Condition Functions

IsRaining

Синтаксис:
CODE
IsRaining

Функция возвращает "0", если текущая погода не классифицируется как дождь. Значение больше "0" и меньше "1" означает процесс смены погоды, например, переход к дождю.
Если возвращается "1", то это означает, что на данной местности устоявшаяся дождливая погода.
Относится к типу: Weather Functions | Condition Functions

IsRidingHorse

Синтаксис:
CODE
[ActorID.]IsRidingHorse

Функция IsRidingHorse возвращает "1", если вызывающий актер (ActorID) в данный момент едет на лошади.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsRunning

Синтаксис:
CODE
[ActorID.]IsRunning

Функция IsRunning возвращает "1", если вызывающий актер (ActorID) в данный момент передвигается (бежит). Если функция вызывается на персонаже игрока и включен режим автопередвижения (клавиша Q), то функция также вернет "1", несмотря на то, на самом ли деле актер передвигается, или же застрял в расщелине и стоит на месте.
См. также: SetForceRun, GetForceRun
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsShieldOut

Синтаксис:
CODE
[ActorID.]IsShieldOut

Функция IsShieldOut возвращает "1", если вызывающий актер (ActorID) в данный момент держит щит.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsSneaking

Синтаксис:
CODE
[ActorID.]IsSneaking

Функция IsSneaking возвращает "1", если вызывающий актер (ActorID) в данный момент находится в режиме "красться".
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsSnowing

Синтаксис:
CODE
IsSnowing

Функция IsSnowing возвратит "0", если текущую погоду нельзя охарактеризовать как снежную (snow).
Значение "1" будет возвращено, если погода на данной местности устоявшаяся и снежная.
Возвращаемое значение, лежащее между 0 и 1, означает процесс смены погоды, например, в сторону снежной.
Относится к типу: Weather Functions | Condition Functions

IsSpellTarget

Синтаксис:
CODE
[ObjectID.]IsSpellTarget MagicID

Пример:
CODE
IsSpellTarget SuperStrengthBuff

Функция IsSpellTarget возвращает "1", если вызывающий объект (ObjectID) в данный момент испытывает влияние указанного в виде параметра (MagicID) магического предмета (заклинания, зачарования или зелья).
См. также: HasMagicEffect
Относится к типу: Magic Functions | Condition Functions | Actor Functions

IsSwimming

Синтаксис:
CODE
[ActorID.]IsSwimming

Функция IsSwimming возвращает "1", если вызывающий актер (ActorID) в данный момент плавает в воде.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsTalking

Синтаксис:
CODE
[ActorID.]IsTalking

Функция IsTalking возвращает "1", если вызывающий актер (ActorID) в данный момент разговаривает.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsTimePassing

Синтаксис:
CODE
[ActorID.]IsTimePassing

Функция IsTimePassing возвращает "1", если вызывающий актер (ActorID) в данный момент спит, отдыхает или путешествует (т.е., время проходит в режиме меню).
Пример:
CODE
if IsTimePassing == 1
; делаем что-то
endif

Эта функция особенно полезна внутри блока MenuMode, когда нужно, чтобы скрипт выполнялся только во время нахождения игрока в режиме меню.
Например, следующий скрипт вернет "1", если игрок на самом деле спит или ожидает (иначе иногда функция MenuMode будет возвращать "1" при отображении меню сна/отдыха даже в том случае, если игрок отменит ее, без сна или отдыха):
CODE
begin MenuMode 1012
if IsTimePassing == 1
; делаем что-то
endif
end
Относится к типу: Player Functions | Condition Functions

IsTorchOut

Синтаксис:
CODE
[NpcID.]IsTorchOut

Функция IsTorchOut возвращает "1", если вызывающий NPC (NpcID) в данный момент вытащил факел.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsTrespassing

Синтаксис:
CODE
[ActorID.]IsTrespassing

Функция IsTrespassing возвращает "1", если вызывающий актер (ActorID) в данный момент нарушает чужую территорию.
Относится к типу: Crime Functions | Condition Functions | Actor Functions

IsTurnArrest

Синтаксис:
CODE
[ActorID.]IsTurnArrest

Функция IsTurnArrest возвращает "1", если вызывающий актер (ActorID) в данный момент говорит со стражем, имея награду за голову. Полезна только как функция условия в приветствии (GREETING), из-за специфики работы функции.
Относится к типу: Player Functions | Condition Functions

IsWaiting

Синтаксис:
CODE
[ActorID.]IsWaiting

Функция IsWaiting возвращает "1", если вызывающий актер (ActorID) в данный момент находится в состоянии ожидания.
См. также: Wait, StopWaiting
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsWeaponOut

Синтаксис:
CODE
[ActorID.]IsWeaponOut

Функция IsWeaponOut возвращает "1", если вызывающий актер (ActorID) в данный момент вытащил оружие.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

IsXBox

Функцией IsXBox возвращается "1", если игра выполняется на платформе XBox. Это требуется из-за специфических особенностей платформы, таких, например, как достижения или управление.
Относится к типу: Miscellaneous Functions

IsYielding

Синтаксис:
CODE
[ActorID.]IsYielding

Функция IsYielding возвращает "1", если вызывающий актер (ActorID) в данный момент получает прибыль.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

K

Kill

Синтаксис:

CODE
[ActorID.]Kill ActorKillerID (optional)

Вызов функции Kill убьет актера ActorID, не затрагивая значение его здоровья, однако важные (essential) для игры квестовые актеры лишь потеряют на время сознание.
Опциональный параметр ActorKillerID является необязательным - он лишь сообщает системе ID актера, ответственного за смерть ("В моей смерти прошу винить Клаву К", помните?). Иногда это требуется в преступлениях, а также при запуске блока OnDeath.
Напомним, что этот тип блока выполняется только один раз, когда указанный актер (ActorKillerID) убивает заскриптованного актера (ActorID). Если параметр не используется, блок выполняется, когда заскриптованный актер умирает.
CODE
begin OnDeath ActorID (не обязательно)
.....
end

Очевидно, что функцию Kill нет смысла размещать внутри блока OnDeath - она должна находиться во внешнем по отношению к нему блоке.
Относится к типу: Statistics Functions | Actor Functions

KillAllActors

Синтаксис:
CODE
KillAllActors

Вызов KillAllActors убивает всех актеров в загруженной зоне. Если вызов происходит во внешней ячейке, то погибнут все актеры на площади 5x5 ячеек вокруг игрока. Если во внутренней ячейке, то все актеры, находящиеся с игроком в этой локации.
Также, как и с функцией Kill, важные (essential) актеры потеряют на время сознание (unconscious, это их обычное «мертвое» состояние).
Использовать функцию нужно крайне осмотрительно…
Относится к типу: Statistics Functions

L

Lock

Синтаксис:
CODE
[DoorID|ContainerID.]Lock lockLevel lockAsOwner (optional)

Пример:
CODE
Lock 30
Lock 75 1

Функция Lock запирает вызываемую дверь (DoorID) или контейнер (ContainerID) до указанного в виде параметра (lockLevel) желаемого уровня сложности. При использовании опционального флага LockAsOwner дверь будет заперта таким образом, как будто ее запер владелец. Это нужно для того, чтобы AI мог определить, является ли проникновение в ячейку за дверью нарушением границ собственности или нет. Если флаг LockAsOwner установлен в "0", то становится неважным, была ли ячейка помечена как "public" (публичная) или "private" (частная собственность).
Несмотря на то, что вы можете ввести любое значение для locklevel, есть набор определенных значений, ассоциированных с каждым уровнем сложности замка. Каждому такому значению соответствует игровая настройка, управляющая значением.

Уровень замка - Значение - Игровая настройка		
Lock Level	Value	Game Setting
Very Easy	7	ILockLevelMaxVeryEasy - Очень легко
Easy	20	ILockLevelMaxEasy - Легко
Average	40	ILockLevelMaxAverage - Средне
Hard	80	ILockLevelMaxHard - Тяжело
Very Hard	99	ILockLevelMaxVeryHard - Очень тяжело
Needs a key	100	- Нужен ключ

См. также: GetLockLevel, GetLocked, Unlock
Относится к типу: Miscellaneous Functions

Look

Синтаксис:
CODE
[ActorID.]Look TargetRefID

Примеры:
CODE
Look UrielSeptimRef

Функция Look принуждает вызывающего актера ActorID следить взглядом за целью TargetRefID. Команда перекроет все обычные события для слежения, пока не будет вызвана функция StopLook или цель не покинет загруженную зону.
См. также: StopLook
Относится к типу: AI Functions | Actor Functions

LoopGroup

Синтаксис:
CODE
LoopGroup GroupName, 2, [flags]

Примеры:
CODE
LoopGroup Idle2
LoopGroup Idle2, 1

Функция LoopGroup проигрывает указанную в виде параметра GroupName анимационную группу. Анимация будет воспроизводиться циклично некоторое целое количество раз (в данном случае 2), после чего последует возврат к анимации бездействия.
Необязательный флаг [flags] может использоваться для проигрывания анимационной группы различными способами.
Флаги:
"0" = Обычный (Normal) - текущая анимация проиграется до конца полного цикла, затем стартует новая анимация со своего стартового фрейма.
"1" = Немедленный запуск (Immediate Start) – Установка флага в "1" приводит к немедленной остановке в текущем фрейме проигрываемой в данный момент анимационной группы, после чего следует старт новой анимации со своего стартового фрейма.
"2" = Немедленный цикл (Immediate Loop) – Установка флага в "2" приводит к немедленной остановке в текущем фрейме проигрываемой в данный момент анимационной группы, после чего стартует новая анимация со своего стартового фрейма и будет воспроизводиться в цикле много раз (по петле).
См. также: AnimGroups, PlayGroup, SkipAnim
Относится к типу: Functions | Animation Functions

M

MenuMode (Function)

Синтаксис:
CODE
MenuMode MenuType (optional)

Пример:
CODE
if (MenuMode == 0)
if (MenuMode 1036 == 1)

Примечание: MenuMode может использоваться как функция или как тип блока.
При вызове без параметра эта функция вернет "1", когда игра находится не в режиме игры, а в режиме отображения меню.
Если указать параметр MenuType явно, то функция вернет "1", когда на экране отобразится указанный тип меню (или указанное конкретное меню).
Типы меню:
1 = "главная четверка" (интерфейс персонажа: статы, магия, инвентарь, квесты)
2 = любое другое меню (сообщения, содержимое контейнера и т.д.)
3 = консоль
Специальные меню:
1001 = Message (Сообщение)
1002 = Inventory (Инвентарь)
1003 = Stats (Статсы)
1004 = HUDMain

1005 = HUDInfo
1006 = HUDReticle
1007 = Loading (Загрузка)
1008 = Container, Barter (Контейнер, обмен)
1009 = Dialog (Диалог)
1010 = HUDSubtitle
1011 = Generic (Обычное)
1012 = SleepWait (Спать/Ждать)
1013 = Pause (Пауза)
1014 = LockPick (Отмычка)
1015 = Options (Опции)
1016 = Quantity
1017 = Audio
1018 = Video
1019 = VideoDisplay
1020 = Gameplay
1021 = Controls (Управление)
1022 = Magic (Магия)
1023 = Map (Карта)
1024 = MagicPopup
1025 = Negotiate (Торговля)
1026 = Book (Книга)
1027 = LevelUp (Уровень)
1028 = Training (Тренировка)
1029 = BirthSign (Знак рождения)
1030 = Class (Класс)
1031 = Attributes (Аттрибуты)
1032 = Skills (Навыки)
1033 = Specilization (Специализация)
1034 = Persuasion (Убеждение)
1035 = Repair (Починка)
1036 = RaceSex (Раса/Пол)
1037 = SpellPurchase (Покупка заклинаний)
1038 = Load (Загрузить)
1039 = Save (Сохранить)
1040 = Alchemy (Алхимия)
1041 = SpellMaking (Создание заклинаний)
1042 = Enchantment (Зачарование)
1043 = EffectSetting
1044 = Main (Главное)
1045 = Breath (Дыхание)
1046 = QuickKeys (Быстрые клавиши)
1047 = Credits (Авторы)
1048 = SigilStone
1049 = Recharge (Перезарядка)
1051 = TextEdit

Относится к типу: Miscellaneous Functions | Condition Functions

Message

Синтаксис:
CODE
Message "Текст сообщения", [var1], [var2], DisplaySeconds

Пример:
CODE
Message "Это сообщение"
Message "У вас есть %.0f часов, чтобы завершить задание", GameHour, 10

Функция Message выводит на экран диалоговое окно с текстом сообщения. Сообщение остается на экране некоторое время, которое задается в виде параметра DisplaySeconds в секундах и должно быть целым числом.
Примечание: Тестирование показало, что этот параметр не влияет на продолжительность отображения сообщения. Если у вас получилось её задать - сообщите нам, пожалуйста!

С помощью Message можно отобразить до 9 переменных. Они показываются в том порядке, в каком перечислены как параметры функции. В сообщении должен быть указан способ отображения переменной.
Отображение переменных

Форматирование чисел:
CODE

Format	Результат
%.2f	Означает, что эта переменная будет отображаться с двумя цифрами после запятой.
%.0f	Формат переменной без цифр после запятой ("0f"), обычно выбирается для отображения целых чисел.
%5.0f	Число перед точкой определяет общую ширину числа. В этом случае всегда будет зарезервировано достаточно места для числа из пяти цифр. Примеры отображения: "Число 12 выиграло" "Число 1234 выиграло"

Переключатели форматирования (Switch)
В Обливионе могут быть использованы следующие переключатели форматирования, которые можно ставить в любой последовательности сразу за символом '%'
CODE

Switch	Результат
"+"	Отображать "+" перед положительными числами
"<Space>"	Оставить пробел перед положительными числами
"-"	Использовать выравнивание по левому краю, а не по правому.
"0"	Заполняющий символ при форматировании будет '0' вместо пробела(' ').

Другие функции
CODE

Format	Результат
%g	Работает как "%.0f", отображая 0 цифр после запятой. Когда длина числа достигает 7 десятичных разрядов (≥ 1000000), игра отобразит его в "научной" записи, например, "1E+006".
%.3e	Отображает все числа в "научной" интерпретации (123000 = 1.23E+005)
%%	Используется для отображения знака процента перед числом.

Примеры:

CODE
Message "Var1:% 5.2f / Var2:% 5.2f" Var1 Var2
"Var1: 123.45 / Var2: -123.45"

CODE
Message "Var1:%05.2f / Var2:%05.2f" Var1 Var2
"Var1: 00123.45 / Var2:-00123.45"

CODE
Message "Var1:%+-5.0f / Var2:%+-5.0f" Var1 Var2
"Var1:+123 / Var2:-123 "

CODE
Message "Var1:% .3e / Var2:% .3e" Var1 Var2
"Var1: 1.234E+2 / Var2:-1.234E+2"

См. также: MessageBox, GetButtonPressed
Относится к типу: Miscellaneous Functions

MessageBox

Синтаксис:
CODE
MessageBox "Сообщение", [var1], ..., [var9], ["button01"], ..., ["button10"]
MessageBox "Сообщение" [var1] ... [var9] ["button01"] ... ["button10"]

Пример:
CODE
MessageBox "Это сообщение"
MessageBox "Начать вращение? Время = %.2f", GameHour, "OK", "Ни за что"
MessageBox "Сколько у вас золота?" "0" "100" "500" "1000" "5000" "10000"

Функция MessageBox отображает сообщение, которое можно отформатировать, а также кнопки, на которые игрок может нажать. MessageBox останавливает время и отображается в виде диалогового окна в центре экрана до тех пор, пока игрок не нажмет на какую-нибудь кнопку. Нажатие можно отследить в скрипте, используя функцию GetButtonPressed.
Может быть задано до 10 кнопок. Если в параметрах функции не задано ни одной, отобразится кнопка "Done".
В MessageBox можно передать до 9 переменных. Они отображаются в том порядке, в каком указаны как параметры функции. В сообщении должен быть указан способ отображения переменной.
Если окно MessageBox открыто, эта функция будет возвращать значение "-1" в функцию GetButtonPressed до тех пор, пока игрок не нажмет на какую-нибудь кнопку.
Форматирование выводимых на экран числовых переменных полностью совпадает с таковым для функции Message. Опции форматирования подробно описаны в ее описании.
См. также: Message, GetButtonPressed
Относится к типу: Miscellaneous Functions

ModActorValue

Синтаксис:
CODE
[ActorID|Player.]ModActorValue StatName value
ModAV StatName value

Пример:
CODE
ModActorValue Strength -10

Функция ModActorValue изменяет указанную в виде параметра (StatName) характеристику вызывающего актера (ActorID) или персонажа игрока (Player) на заданное значение (value), не затрагивая базового значения характеристики (в дополнение к повреждениям и магическим изменениям).
Примечания:
Функция ModActorValue изменяет характеристику на заданное значение, не затрагивая базового значения характеристики. Изменение при помощи ModActorValue может превышать максимум в 100 при изменении навыка и характеристики, измененный параметр будет показан красным (damaged - повреждение) или зеленым (restored - восстановление), чтобы показать временную модификацию. Неизмененный параметр будет отображаться синим цветом. Основное использование Modactorvalue — это проклятья или благословения, которые нельзя развеять, и заклинания и магические предметы, модифицирующие навык/характеристику на значение, больше 100.
Если вы хотите сделать в скрипте постоянные изменения для актера, не используйте ModActorValue. Вместо этого нужно использовать SetActorValue или применить заклинание, которое изменит характеристику до нужного вам значения. Хороший способ — создать Способность (Ability) и использовать AddSpell, чтобы добавить ее в список заклинаний.
Если вы используете ModActorValue в скрипте, то меняется скриптовый модификатор (Script Modifier), и ТОЛЬКО скрипт может вернуть его назад. Другими словами, если есть скрипт
CODE
player.modav health 100

вы также должны сделать
CODE
player.modav health -100

иначе эти 100 очков *навсегда* останутся в Script modifier.
Хороший пример имеется в скрипте Обливиона DarkScalesScript.
Для временных изменений, которые могут восстанавливаться заклинанием, вы можете использовать OBSE-функцию ModActorValue2 .
Консоль:
ModActorValue действует иначе при использовании в консоли. В консоли отрицательные значения изменяют магический модификатор (Magic Modifier) и может быть скорректировано Заклинанием-Восстановлением, тогда как положительные значения изменяют только игровым модификатором (Game Modifier) максимум до нуля.

Причина этих различий в поведении ModActorValue в консоли - использование команды для целей теста, тогда как в скриптах она используется для временных эффектов, которые не могут быть изменены заклинаниями или внутренней системой игры.
Практика

Основы:
Когда вы (или игра) использует GetActorValue, вы получаете сумму базового значения актера плюс трех модификаторов:
Game Modifier используется для "постоянного" эффекта, типа Damage и Restore.
Magic Modifier используется для "временного", развеваемого магического эффекта, такого как Drain и Fortify.
Script Modifier используется для "временного" неразвеваемого скриптового эффекта, такого как благословения и проклятия.
Функции ModActorValue и ForceActorValue изменяют только модификатор скрипта.

Изменения, сделанные этими функциями, не могут быть восстановлены внутриигровыми средствами, подобно естественному восстановлению здоровья или магии. Для этого вам нужно "уничтожить" их в скрипте.

В скрипте эти функции ведут себя, как описано выше, в консоли же они ведут себя несколько иначе в связи с тем, что они использовались Bethesda для целевого тестирования.
См. также: Stats List, GetActorValue, GetBaseActorValue, SetActorValue, ForceActorValue, ModActorValue2 (OBSE)
Относится к типу: Actor Value Functions | Statistics Functions | Actor Functions

ModAmountSoldStolen

Синтаксис:
CODE
ModAmountSoldStolen iValue

Пример:
CODE
ModAmountSoldStolen 200

Функция ModAmountSoldStolen изменяет сохраненное значение общей стоимости проданных игроком украденных вещей на указанное в виде параметра iValue значение.
См. также: GetAmountSoldStolen
Относится к типу: Player Functions | Condition Functions

ModBarterGold

Синтаксис:
CODE
ModBarterGold float

Пример:
CODE
ModBarterGold -20.0

Функция ModBarterGold изменяет количество золота для торговли на указанное в виде параметра значение (float).
См. также: SetBarterGold
Относится к типу: Statistics Functions | Actor Functions

ModCrimeGold

Синтаксис:
CODE
[ActorID.]ModCrimeGold float

Пример:
CODE
ModCrimeGold -20.0

Функция ModCrimeGold изменяет текущее значение криминального золота у вызывающего актера (ActorID) на заданное значение (float).
См. также: GetCrimeGold, SetCrimeGold
Относится к типу: Crime Functions | Actor Functions

ModDisposition

Синтаксис:
CODE

[ActorID.]ModDisposition ActorRefID float

Пример:
CODE
ModDisposition player -20

Функция ModDisposition позволяет изменить отношение вызывающего актера (ActorID) к указанному в виде параметра ActorRefID персонажу на указанное значение (float).
См. также: GetDisposition
Относится к типу: Statistics Functions | Actor Functions

ModFactionRank

Синтаксис:
CODE
[ActorID.]ModFactionRank FactionID, modValue

Пример:
CODE
ModFactionRank FightersGuild, 1

Функция ModFactionRank изменяет ранг вызывающего актера (ActorID) в указанной в виде параметра фракции FactionID на требуемое значение modValue. Отметьте, что данная функция эффекта не принесет, если актер не является членом указанной фракции, а также то, что ранг никогда не опустится ниже 0.
Относится к типу: Faction Functions | Actor Functions

ModFactionReaction

Синтаксис:
CODE
ModFactionReaction FactionID, TargetFactionID, ModValue

Пример:
CODE
ModFactionReaction FightersGuild playerFaction -20

Функция ModFactionReaction изменяет отношение указанной в виде параметра фракции (FactionID) ко второй указанной фракции (TargetFactionID) на значение ModValue.
Смотрите раздел "Фракции", чтобы узнать больше.
См. также: SetFactionReaction, GetFactionReaction
Относится к типу: Faction Functions

ModPCAttribute

Синтаксис:
CODE
ModPCAttribute StatName, Value

Пример:
CODE
ModPCAttribute Strength 1

Функция ModPCAttribute позволяет дать персонажу игрока прирост характеристики StatName на указанное значение Value, которое будет считаться в игре как "постоянное" (также, как вы увеличиваете атрибуты (attribute) при повышении уровня). По сравнению с функцией ModPCSkill в данном случае Value не обязательно должно быть положительным.
См. также: Stats List, ModPCSkill
Относится к типу: Player Functions

ModPCFame

Синтаксис:
CODE
ModPCFame value

Пример:
CODE
ModPCFame 4

Функция ModPCFame добавляет значение к текущей положительной известности персонажа. Отметьте, что только у игрока может быть как положительное, так и отрицательное значения популярности.
Относится к типу - Player Functions

ModPCInfamy

Синтаксис:
CODE
ModPCInfamy value

Пример:
CODE
ModPCInfamy 3

Функция ModPCInfamy добавляет значение к текущей отрицательной известности персонажа игрока. Отметим, что только у игрока есть как положительное, так и отрицательное значения популярности.
Относится к типу - Player Functions

ModPCMiscStat

Синтаксис:
CODE
ModPCMiscStat MiscStatID Value

Пример:
CODE
ModPCMiscStat 12 1; изменяет статистику "сваренных зелий"

Функция ModPCMiscStat позволяет изменять любую запись (MiscStatID) из "неосновных" характеристик персонажа игрока на указанное значение Value.

Перечень MiscStatID:

0 DAYS IN PRISON
1 DAYS PASSED
2 SKILL INCREASES
3 TRAINING SESSIONS
4 LARGEST BOUNTY
5 CREATURES KILLED
6 PEOPLE KILLED
7 PLACES DISCOVERED
8 LOCKS PICKED
9 PICKS BROKEN
10 SOULS TRAPPED
11 INGREDIENTS EATEN
12 POTIONS MADE
13 OBLIVION GATES SHUT
14 HORSES OWNED
15 HOUSES OWNED
16 STORES INVESTED IN
17 BOOKS READ
18 SKILL BOOKS READ
19 ARTIFACTS FOUND
20 HOURS SLEPT
21 HOURS WAITED
22 DAYS AS A VAMPIRE
23 LAST DAY AS VAMPIRE
24 PEOPLE FED ON
25 JOKES TOLD
26 DISEASES CONTRACTED
27 NIRNROOTS FOUND
28 ITEMS STOLEN
29 ITEMS PICKPOCKETED
30 TRESPASSES
31 ASSAULTS
32 MURDERS
33 HORSES STOLEN

См. также: GetPCMiscStat, ModPCSkill, ModPCAttribute
Относится к типу - Player Functions

ModPCSkill

Синтаксис:
CODE
ModPCSkill SkillName Amount

Пример:
CODE
ModPCSkill Block 1

Функция ModPCSkill увеличивает указанный в виде параметра навык SkillName на требуемое значение Amount. Удостоверьтесь, что используется положительное число. Увеличение, по отношению к набору уровней, считается точно также, как и при обычном «наборе» навыка.
Чтобы понизить навык или поднять его без изменения уровня, используйте Player.SetActorValue.

Примечание: В Beta Patch 1.1, вызов функции был исправлен, теперь он увеличивает уровень и устанавливает значение использования навыка в 0.0. Это исправило ситуацию с отрицательными значениями, но при вызове функции, когда вы близки к поднятию уровня, навык увеличится на значение, необходимое для поднятия уровня.

Примечание: В непропатченной игре ModPCSkill увеличит ваш уровень навыка на "1" и уменьшит значение использования навыка на требуемое значение для поднятия уровня. Если было 2.56/7.68 до вызова ModPCSkill, то после станет -5.12/(New Level Max). Повторные вызовы продолжат уменьшать значение использования навыка на каждый (New Level Max). Если вы уже близко к набору уровня, отрицательное значение будет маленьким и быстро уйдет, но если добавить больше одного уровня навыка, или вы часто вызываете ModPCSkill (или AdvancePCSkill), то такую нехватку придется долго компенсировать и уровень развития «застрянет».
См. также: Stats List, SetActorValue, ModPCAttribute
Относится к типу: Player Functions

ModScale

Синтаксис:
CODE
[ObjectID.]ModScale Value

Пример:
CODE
ModScale 0.4

Функция ModScale увеличивает масштабный множитель, на который умножается размер вызывающего объекта (ObjectID), на указанное в виде параметра значение Value (переменная типа float). Функция строго прибавляющая — если текущий масштаб равен "1.2" и вы вызываете ModScale .5, новое значение масштабного множителя будет равно 1.7.
См. также: SetScale
Относится к типу: Statistics Functions

MoveTo

Синтаксис:
CODE
[ActorID|Player.]MoveTo MarkerID, x, y, z (optional)

Пример:
CODE
MoveTo HiddenCaveMarker
MoveTo player, 512, 0, 0

Функция MoveTo перемещает вызывающего актера (ActorID) или персонажа игрока к местоположению указанного в виде параметра объекта (MarkerID). Параметры x, y, z — опциональное смещение в игровых единицах измерения (units) от указанного объекта.
Примечания:
Если функция используется для перемещения игрока, она действует как функция Return — следующие за ней строки в скрипте отработаны не будут.
Функция работает только с актерами. Использование ее для перемещения других типов объектов, например, контейнеров и активаторов, приведет только к обновлению координат объекта, но в игровом мире - нет.
Дополнительные строки в скрипте могут быть необходимы, чтобы гарантировать корректное перемещение объекта:
CODE

myObject.disable
myObject.moveTo [location]
myObject.enable
set xp to myObject.getPos x
myObject.setPos x xp
Скриптовые функции, в которых одни актеры являются целью для других (подобно SayTo или StartCombat), работать не будут, если функция MoveTo используется на той же цели и в том же фрейме с этими функциями, даже если цель и переместится в сторону на несколько дюймов.
Прим. Vitalka:
Некоторое дополнение по функциям MoveTo и MoveToMarker.
Если эти функции используются на игроке, то после них обязательно должна стоять функция Return. Причем между ними нельзя ставить никакие команды.
Так, например, нельзя:
CODE
Player.MoveTo HiddenCaveMarker
Set a to b
Return

А так можно:
CODE
Set a to b
Player.MoveTo HiddenCaveMarker
Return

Иначе появляются страшные "глюки". Во всяком случае, мне удалось выйти из положения именно таким образом.
См. также: PositionWorld, PositionCell
Относится к типу - Movement Functions

MoveToMarker

Синтаксис:
CODE
[ObjectID.]MoveToMarker MarkerID

Пример:
CODE
MoveToMarker MarkerX
MoveToMarker FG04A

Функция MoveToMarker используется для перемещения вызывающего объекта (ObjectID) в указанное в виде параметра место, помеченное маркером (MarkerID). Заметьте, что "MarkerID" указывает на ID копии базового маркера (Reference Editor ID), размещенного в игровом мире. Его можно найти на соответствующей вкладке окна Reference Windows.
См. также примечания Vitalka в описании функции MoveTo...
См. также: PositionWorld, PositionCell
Относится к типу - Movement Functions

P

PayFine

Синтаксис:
CODE
PayFine

Функция PayFine изымает из инвентаря игрока количество "криминального" золота, сворованного им, а также конфискует все украденные предметы, очищая таким образом инвентарь от всего "нажитого непосильным трудом".

Примечания:
Тюрьма содержит пару связанных дверей-маркеров ("Prisonmarker"). Один маркер установлен непосредственно в интерьере тюремной камеры, а второй - во внешней локации, куда персонаж игрока будет перемещен после отбывания наказания. Конфискованные у игрока ворованные предметы помещены в ящик "Ворованные предметы" (Stolengoods), который находится в той же внутренней ячейке тюрьмы, что и маркер (Prisonmarker).
См. также: GoToJail, PayFineThief
Относится к типу: Crime Functions

PayFineThief

Синтаксис:

CODE
PayFineThief

Функция PayFineThief изымает в виде штрафа все "криминальное" золото, сворованное игроком.
См. также: PayFine
Относится к типу: Crime Functions

PickIdle

Синтаксис:
CODE
[ActorID.]PickIdle

Функция PickIdle заставляет вызывающего актера (ActorID) выбрать новое анимационное движение (idle).
Относится к типу: Animation Functions | Actor Functions

PlaceAtMe

Синтаксис:
CODE
[ObjectID.]PlaceAtMe ItemID, count, [distance], [direction]

Пример:
CODE
player.PlaceAtMe NinjaMan, 1, 256, 1

Функция PlaceAtMe помещает указанный в виде параметра объект (ItemID) рядом с вызывающим объектом (ObjectID) на указанном расстоянии [distance] и в определенном направлении [direction]. Если место перемещения небезопасно (в воздухе, в стене, и т.д.), объект будет помещен на одной из оставшихся осей или прямо на месте объекта.
Направления:
CODE
0 = впереди
1 = сзади
2 = слева
3 = справа
Эта функция может использоваться с существами из уровневого списка.
Однако она не может использоваться с уровневыми предметами. Уровневые предметы не предназначены для размещения в мире — их использование ограничивается контейнерами. Попытка таким путем создать объект из уровневого списка приведет лишь к появлению желтого восклицательного знака (Marker_Error.NIF).

Внимание: функции Disable и Remove не одно и то же!
Отметьте, что на самом деле нет ясности с перемещенными функцией PlaceAtMe объектами - они отключены в прежней локации или нет? И очистит ли их новая загрузка сохранения? Судя по тестам, это не так, и это может быть серьезной проблемой, например, с заклинаниями, которые могут создать массу активаторов, но ведь после этого их нужно отключать с помощью функции Disable!
Таким образом, кажется вполне целесообразным использование для целей перемещения функцию MoveTo на устойчивых копиях объектов в тех случаях, где это возможно, вместо пары PlaceAtMe / Disable
В то же самое время MoveTo может оказаться такой же удобной, как и функция PlaceAtMe. Однако, если вам действительно нужно использовать именно PlaceAtMe в вашем моде, то будет справедливо, если вы укажете на этот факт в Readme-файле вашего мода.

Получение ссылки на созданный объект
При использовании одного объекта эта функция вернет ссылку на созданный объект, так что это может использоваться с дополнительным вызовом функций.
Например:
CODE
scn scriptName
ref refName
begin blockName
set refName to refCreatingObject.PlaceAtMe ObjectToBeCreated 1, 0, 0
end

Переменная refName теперь содержит ссылку на ObjectToBeCreated.
Эта ссылка верна только для объектов, которые нельзя поместить в инвентарь. Доступ по ссылке на предмет инвентаря после того, как кто-то его поднял, может вызвать CTD.

Использование в консоли
При использовании PlaceAtMe в консоли нужно использовать нужный FormID предмета, а не его EditorID.

Поэтому, чтобы добавить себе отмычку, вместо использования
CODE
player.PlaceAtMe lockpick 1, 256, 0

наберите
CODE
player.PlaceAtMe 00000A 1, 256, 0

FormID можно найти в конструкторе в свойствах объекта справа от колонки EditorID. Колонка с FormID спрятана и ее требуется развернуть. Использовать этот код в консоли следует для всех объектов - как предметов, так и актеров.
Относится к типу: Miscellaneous Functions

PlayBink

Синтаксис:
CODE
PlayBink filename.bik AllowEscapeFlag (optional)

Примеры:
CODE
PlayBink "MyNew.bik" (проигрывание Oblivion\data\video\MyNew.bik)
PlayBink "Mymod\MyNew.bik" (проигрывание Oblivion\data\video\Mymod\MyNew.bik)
Playbink "..\sound\videos\Mynew.bik" (проигрывание Oblivion\sound\videos\MyNew.bik)

Функция PlayBink позволяет проигрывать пользовательские видеофайлы в формате filename.bik.
Примечания:
Ваши файлы в формате *.bik (Bink files) должны находиться на вашем жестком диске и они не должны быть упакованы в файле BSA.
Имя файла нужно указывать в кавычках.
Путь к файлу должен быть следующим: Oblivion\Data\Video\
Если флаг AllowEscapeFlag не равен "0", то пользователь может прервать проигрывание файла *.bink при нажатии на клавишу ESC. В противном случае вам придется просматривать видео до конца.
Относится к типу: Miscellaneous Functions

PlayGroup

Синтаксис:
CODE
PlayGroup GroupName, [flags]

Примеры:
CODE
PlayGroup Walk
PlayGroup Walk, 1

Функция PlayGroup проигрывает анимационную группу, определенную в GroupName.
Необязательный флаг [flags] может использоваться, чтобы начать прогрывание анимационной группы различными способами.
Флаги:
0 = Обычный (Normal) - текущая анимация проиграется до конца полного цикла, затем начнется проигрывание новой анимации с самого начала, со своего первого стартового фрейма.
1 = Немедленный старт (Immediate Start) – Установка флага в единицу приведет к немедленной остановке в текущем фрейме проигрываемой в данный момент анимационной группы, после чего начнется проигрывание новой анимации со своего стартового фрейма.
2 = Немедленный запуск цикла (петли) (Immediate Loop) – Установка флага в 2 приводит к немедленному прекращению проигрывания в текущем фрейме анимационной группы, после чего стартует новая анимация со своего стартового фрейма и будет воспроизводиться в цикле много раз (по петле).
Примечание:
Проигрывание этой функцией некоторых анимационных групп на персонаже игрока может его "заморозить" и он перестанет реагировать на ваши действия. Вернуть его в нормальное состояние поможет вызов функции PickIdle или вызов PlayGroup со следующими параметрами:
CODE
PlayGroup Idle, 1

Относится к типу: Animation Functions

PlayMagicEffectVisuals

Синтаксис:
CODE
[ObjectID.]PlayMagicEffectVisuals MagicEffectID, Duration (optional)

Пример:
CODE
PlayMagicEffectVisuals FIDG
pme FTHR 10

Функция PlayMagicEffectVisuals накладывает визуальную часть магического эффекта на вызывающем объекте (ObjectID). Визуализация накладывается, по крайней мере, на один цикл, вне зависимости от параметра duration (продолжительность), а если продолжительность не указана, то будет отображаться бесконечно или пока не окончится проигрывание анимации. Заметьте, что для этой функции duration указывать не обязательно, она проиграется хотя бы раз, если вы не указали большую продолжительность.
Прим. ZomBoss:
Ф-я не дала результата в консоли.
Попытался задействовать этот проигрыватель эффектов. Нужно было поджечь дерево... Эффект нулевой.
CODE
PreloadMagicEffect FIDG
PlayMagicEffectVisuals FIDG 10
Когда я "прикрутил" на непися скрипт с функцией
CODE
PlayMagicEffectVisuals FIDG 10
то тоже никакого эффекта не увидел.
См. также: PlayMagicShaderVisuals, StopMagicEffectVisuals
Относится к типу: Magic Functions

PlayMagicShaderVisuals

Синтаксис:
CODE
PlayMagicShaderVisuals EffectShaderID, Duration (optional)

Пример:
CODE
PlayMagicShaderVisuals GhostEffect
pms effectShockShield 10

Функция PlayMagicShaderVisuals накладывает визуализацию шейдерного эффекта на вызывающем объекте. Визуализация накладывается, по крайней мере, на один цикл, вне зависимости от параметра duration (продолжительность), а если продолжительность не указана, то будет показываться бесконечно или пока не окончится проигрывание анимации.
Прим. ZomBoss:
При размещении на объекте
CODE
PlayMagicShaderVisuals effectFireDamage 10

отобразились какие-то мелкие и редкие чуть заметные огненные шарики, которые появлялись в стороне от объекта.
См. также: PlayMagicEffectVisuals, StopMagicShaderVisuals
Относится к типу: Magic Functions

PlaySound

Синтаксис:
CODE
PlaySound SoundID

Пример:
CODE
PlaySound SPLIllusionCast

Функция PlaySound проигрывает указанный в виде параметра SoundID звук, без определенного места звучания.
Примечание: Звуки для этой функции определяются в разделе КС Miscellaneous/Sounds.
См. также: PlaySound3D, StreamMusic
Относится к типу: Miscellaneous Functions

PlaySound3D

Синтаксис:

CODE
[ObjectID.]PlaySound3D SoundID

Пример:
CODE
PlaySound3D SPLIllusionCast

Функция PlaySound3D проигрывает указанный в виде параметра SoundID звук, исходящий из точки нахождения вызывающего объекта ObjectID.
Примечание: Звуки для этой функции определяются в разделе конструктора Miscellaneous/Sounds.
См. также: PlaySound, StreamMusic
Относится к типу: Miscellaneous Functions

PositionCell

Синтаксис:
CODE
[ObjectID.]PositionCell x, y, z, zRot, CellID
poscell x, y, z, zRot, CellID

Пример:
CODE
PositionCell 76345, 100.56, 215, 176, ScaryCavern

Функция PositionCell переносит вызывающий объект (ObjectID) во внутреннюю ячейку CellID с координатами x, y, z. Объект будет повернут на угол, определенный в параметре zRot. Все параметры - вещественные переменные.
Примечание: Почти всегда вы будете использовать функцию MoveTo вместо данной команды.
Относится к типу: Movement Functions

PositionWorld

Синтаксис:
CODE
[ObjectID.]PositionWorld x, y, z, zRot, WorldID
PosWorld x, y, z, zRot, WorldID

Пример:
CODE
PositionWorld 76345, 100.56, 215, 176, Tamriel

Функция PositionWorld переносит объект (ObjectID) во внешнюю ячейку с координатами x, y, z в указанное игровое пространство мира. Объект будет повернут на угол, определенный в параметре zRot. Все параметры - вещественные переменные.
Примечание: Почти всегда вы будете использовать MoveTo вместо данной функции.
Относится к типу: Movement Functions

PreloadMagicEffect

Синтаксис:
CODE
PreloadMagicEffect EffectID

Пример:
CODE
PreloadMagicEffect BACU

Функция PreloadMagicEffect подгрузит частицы для магического эффекта, если они еще не загружены. Функция не делает ничего, если они уже в памяти. Используется только тогда, когда кастует заклинание не-актер, так как актеры подгружают свои заклинания автоматически.
Относится к типу: Magic Functions

R

RefreshTopicList

Синтаксис:
CODE
RefreshTopicList

Функция RefreshTopicList позволяет вручную обновить список тем диалога. Обычно этого делать не требуется, но бывает полезным в некоторых случаях (как правило, если скрипт работает в режиме вывода на экран различных меню (MenuMode).
Относится к типу: Dialogue Functions

ReleaseWeatherOverride

Синтаксис:
CODE
ReleaseWeatherOverride

Функция ReleaseWeatherOverride позволяет погоде меняться в нормальном динамическом ритме. Используется непосредственно после вызова функций SetWeather или ForceWeather с установленным флагом WeatherOverrideFlag.
См. также: SetWeather, ForceWeather
Относится к типу: Weather Functions | Condition Functions

RemoveAllItems

Синтаксис:
CODE
[ActorID|Player]RemoveAllItems TargetContainerID (optional), RetainOwnershipFlag (optional)

Пример:
CODE
RemoveAllItems
RemoveAllItems TreasureChest
RemoveAllItems FriendlyJailer, 1

Функция RemoveAllItems удаляет все предметы из инвентаря вызывающего актера (ActorID) или персонажа игрока. Если указан необязательный параметр - целевой контейнер или инвентарь (TargetContainerID), то объекты перемещаются в него, иначе они просто уничтожаются. Если указать флаг RetainOwnershipFlag = 1, то предметы остаются в собственности старых владельцев, иначе владение очищается.
Отметьте, что квестовые предметы этой функцией не перемещаются в инвентарь игрока, однако, если перемещение производится из контейнера в контейнер, то никаких ограничений для этой операции нет.
См. также: RemoveItem
Относится к типу: Object Functions

RemoveFlames

Синтаксис:
CODE
[ObjectID.]RemoveFlames

Функция RemoveFlames удаляет объект пламени FlameNode с вызывающего объекта ObjectID.


См. также: FlameNode, CanHaveFlames, HasFlames, AddFlames
Относится к типу: Object Functions

RemoveItem

Синтаксис:
CODE
[ActorID|Player.]RemoveItem ObjectID, Count

Пример:
CODE
RemoveItem Gold001 50

CODE
Ref MyItem
Short count
set MyItem to ArenaAkaviriLongSword
set count to 1
player.removeitem MyItem Count

Функция RemoveItem удаляет из инвентаря вызывающего актера (ActorID) указанный в виде параметра объект (ObjectID) в количестве Count.
Примечания:
В качестве ObjectID допустимо использовать ref-переменную (reference variable), а тип переменной для Count - как short (короткая целочисленная).
Если ObjectID относится к уровневому списку, то будет удален уровневый предмет, сгенерированный из него.
См. также: AddItem, GetItemCount, RemoveAllItems
Относится к типу: Object Functions

RemoveMe

Синтаксис:
CODE
[ItemID.]RemoveMe TargetContainerID (optional)

Пример:
CODE
RemoveMe
RemoveMe player

Вызов функции RemoveMe позволяет удалять вызывающий объект (ItemID) из инвентаря или контейнера, в котором он находится (если это возможно). Если указан необязательный параметр - целевой инвентарь или контейнер (TargetContainerID), то объект перемещается в него.
Примечания:
Эта функция действует при вызове аналогично функции "Return" – т.е., строки скрипта, следующие непосредственно за ней, выполняться НЕ БУДУТ (так как объект просто уничтожает сам себя в процессе удаления из инвентаря).
Эта функция не выводит на экран сообщение "xxx был удален"
Когда вы используете функцию RemoveMe сразу же после того, как объект был добавлен в инвентарь, то такая операция может вызвать различные непредсказуемые вылеты из игры. Для предупреждения таких ситуаций есть простой способ - установить задержку для выполнения функции RemoveMe на время, превышающие один фрейм. Установив счетчик на 10, вы задержите выполнение функции RemoveMe, добавив один шаг при каждом вызове режимов MenuMode или GameMode, и достигнете таким образом необходимой цели.
Когда функция RemoveMe вызывается в меню обмена (barter-menu; например, в пределах блока OnAdd), могут происходить вылеты, когда предмет в скрипте суммирующийся и вы продаете более чем один из них, а затем переключаетесь с продажи на покупку. Чтобы это предотвратить, перед удалением предмета следует или дождаться закрытия диалогового окна меню обмена (Menumode=1009), или не использовать такой скрипт вообще в отношении суммирующихся неуникальных предметов (стрел, например).

См. также: DropMe
Относится к типу: Object Functions

RemoveScriptPackage

Синтаксис:
CODE
[ActorID.]RemoveScriptPackage

Примеры: (Из скрипта MS45DarMaScript)
CODE
if getiscurrentPackage MS45DarMaPrisonerGreetPlayer == 1
If getinsamecell player == 0
removescriptpackage
endif

endif

Функция RemoveScriptPackage исключает любой текущий выполняемый скриптовый пакет для вызывающего актера (ActorID). Это бывает иногда необходимо, если вы добавили скриптовый пакет, который был вызван для бессрочного выполнения.
Примечание: Удаление скриптового пакета в случаях, когда у вызывающего актера скриптовых пакетов нет, может привести к прекращению работы скриптовых пакетов у всех копий данного персонажа, размещенных в игровом мире. Поэтому использовать данную функцию следует на актерах, у которых определены пакеты AI с одним или несколькими условиями.
Смотрите также: AddScriptPackage
Относится к типу - AI Functions

RemoveSpell

Синтаксис:
CODE
[ActorID.]RemoveSpell SpellID

Пример:
CODE
RemoveSpell DisTickleBritch
CODE
If Removespell WeirdAbility == 0; Toggle Ability on/off
Addspell WeirdAbility
Endif

Функция RemoveSpell удаляет указанное заклинание SpellID из списка заклинаний вызывающего актера (ActorID). Функция возвращает "1", если заклинание успешно удалено, и "0" — если заклинание не может быть удалено (например, его просто нет в списке).
См. также: AddSpell
Относится к типу: Magic Functions

Reset3DState

Синтаксис:
CODE
Reset3DState

Функция Reset3DState очищает любые сохраненные данные об анимации/физике объекта. Используется, в основном, для сброса ловушек после их использования.
Относится к типу: Miscellaneous Functions

ResetFallDamageTimer

Синтаксис:
CODE
[ObjectID.]ResetFallDamageTimer

Функция ResetFallDamageTimer сбрасывает таймер падения. При сбросе таймера вызывающий объект (ObjectID) получает повреждения при падении так, как будто он начал падать только что.
Относится к типу: Miscellaneous Functions

ResetHealth

Синтаксис:
CODE
[ActorID.]ResetHealth

Функция ResetHealth устанавливает здоровье вызывающего актера (ActorID) к его базовому значению (полностью здоров).
Относится к типу: Statistics Functions

ResetInterior

Синтаксис:
CODE
ResetInterior CellID

Пример:
CODE

ResetInterior MyDungeon01

Функция ResetInterior помечает ячейку как истекшую (expired, т.е. давно не посещаемую), поэтому она при следующем вызове будет загружена очищенной, как будто прошло несколько дней.
Игрок не может находиться в этой ячейке при выполнении этой команды.
Относится к типу: Miscellaneous Functions

Resurrect

Синтаксис:
CODE
[ActorID.]Resurrect AnimateFlag (optional)

Функция Resurrect воскрешает вызывающего актера ActorID. Если опциональный флаг AnimateFlag = 1, то воскрешение актера будет анимировано (поднимающийся, как будто из нокдауна). В противном случае он просто появится уже воскрешенным в своем обычном состоянии.
Примечание: В некоторых ситуациях флаг необходим, чтобы избежать обрушения игры.
Пример скрипта:
CODE
short FlagDead; Устанавливается в "1", когда актер мёртв, и в "0" когда анимация воскрешения завершена.

short ResurrectAminTimer; Считает фреймы с момента начала воскрешения

begin gamemode

; Устанавливаем флаг смерти персонажа в "1" в момент смерти.
; (Примечание: Вы, возможно, имели бы здесь текущий пакет AI)
if (Actor.GetDead == 1)
set FlagDead to 1
endif

; Таймер, который предотвратит запуск скрипта до того, как
; анимация воскрешения завершится (включая пакеты AI).
; Стартует при воскрешении.
if (FlagDead==1) && (Actor.GetDead ==0)
set ResurrectAminTimer to ResurrectAminTimer+1
endif

; Через 200 фреймов (достаточно для завершения анимации)
; возобновляем работу скрипта.
if ResurrectAminTimer > 200
set FlagDead to 0
set ResurrectAminTimer to 0
Actor.resurrect 1
endif

; Когда персонаж жив и анимация завершена, запускаем главный скрипт
if FlagDead == 0
; Заполните здесь требуемые действия для актера
endif

Относится к типу: Statistics Functions | Actor Functions

Rotate

Синтаксис:
CODE
[ObjectID.]Rotate axis, degrees/sec

Пример:
CODE
Rotate x, 100

Функция Rotate вращает вызывающий объект (ObjectID) по выбранной оси axis (x, y или z) с указанной скоростью вращения (degrees/sec). Единица измерения скорости - градусы в сек. Движение основано на местном вращении объекта. Таким образом, положительное движение по оси "y" будет вращать объект по его местному вектору, направленному вперед.

Примечание: Из-за метода, который используется, данная функция непригодна для сложного вращения: если вы хотите вращать объект, который уже вращается, то функция может работать не так, как вы ожидаете. Для повторяющегося или сложного вращения нужно использовать функции анимации - они и быстрее, и надежнее.
Относится к типу: Movement Functions

S

SameFaction

Синтаксис:
CODE
[ActorID.]SameFaction TargetActorID

Функция SameFaction возвращает "1", если вызывающий актер (ActorID) числится в той же фракции, что и указанный целевой актер (TargetActorID), т.е. вместе они состоят хотя бы в одной общей фракции.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

SameFactionAsPC

Синтаксис:
CODE
[ActorID.]SameFactionAsPC

Функция SameFactionAsPC возвращает "1", если вызывающий актер (ActorID) находится в той же фракции, что и игрок. Не работает на объектах.

Относится к типу: Actor State Functions | Condition Functions | Actor Functions

SameRace

Синтаксис:
CODE
[ActorID.]SameRace TargetActorID

Функция SameRace возвращает "1", если вызывающий актер (ActorID) той же расы, что и указанный в виде параметра целевой актер TargetActorID.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

SameRaceAsPC

Синтаксис:
CODE
[ActorID.]SameRaceAsPC

Функция SameRaceAsPC возвращает "1", если вызывающий актер (ActorID) той же расы, что и персонаж игрока.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

SameSex

Синтаксис:
CODE
[ActorID.]SameSex TargetActorID

Функция SameSex возвращает "1", если вызывающий актер (ActorID) того же пола, что и указанный в качестве параметра целевой актер (TargetActorID).
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

SameSexAsPC

Синтаксис:
CODE
[ActorID.]SameSexAsPC

Функция SameSexAsPC возвращает "1", если вызывающий актер (ActorID) того же пола, что и персонаж игрока.
Относится к типу: Actor State Functions | Condition Functions | Actor Functions

Say

Синтаксис:
CODE
[ActorID|ObjectID.]Say TopicID [ForceSubtitleFlag] [SpeakerID] [UnknownFlag]

Пример:
CODE
Say SecretTopic
Say DAMolagBalSpeech 1 DAMolagBalVoice 1

Функция Say может использоваться для того, чтобы NPC или объект проговорил один раз реплику SpeakerID из темы TopicID.
Примечания:
Губы NPC будут синхронизированы с его речью, но это не вызовет прерывания в выполнении текущего пакета AI.
Функция возвращает время в секундах, необходимое NPC для того, чтобы проговорить весь текст до конца. Это может быть полезно для создания последовательных действий.
Если необязательный флаг ForceSubtitleFlag = 1, то субтитры будут отображаться вне зависимости от расстояния между говорящим и игроком.
Используя необязательный параметр SpeakerID, можно заставить произнести что-нибудь существо (creature), активатор или другой объект. ID используется при определении расы и пола воспроизводимого голоса, т.е., для выбора подходящего звукового файла. При этом NPC не обязательно должен быть размещён в мире.
Флаг UnknownFlag следует установить в "1" во всех скриптах, связанных с даэдрическими святынями. Рекомендуется также устанавливать его в "1" всякий раз, когда SpeakerID указывается явно в качестве параметра.
См. также: SayTo
Относится к типу: Dialogue Functions | Actor Functions

SayTo

Синтаксис:
CODE
[ActorID.]SayTo TargetActorID TopicID ForceSubtitleFlag (optional)

Пример:
CODE
SayTo Joe SecretTopic
SayTo player SecretTopic 1

Используйте функцию SayTo, чтобы вызывающий NPC (ActorID) сказал одну реплику (TopicID) игроку или другому NPC (TargetActorID). Говорящий NPC будет следить за целью поворотом головы или поворачиваться к ней, если это позволяет текущий пакет, но не будет ждать, чтобы приблизиться на нужное расстояние (поэтому убедитесь, что функция вызывается, когда расстояние для разговора достаточно).
Функция возвращает время в секундах, необходимое NPC для того, чтобы проговорить весь текст. Это может быть полезно при создании последовательных действий.
Если флаг ForceSubtitleFlag = 1, субтитры будут отображаться вне зависимости от расстояния между говорящим и игроком.
Пример скрипта:
CODE
begin gamemode
if timer > 0
set timer to timer - getsecondspassed
elseif talk == 1
set lastStage to CharacterGen.tauntStage
set timer to SayTo player, CharGenTaunt2 1
if getstage characterGen == 9
set characterGen.convTimer to timer - .5
endif
endif
end

См. также: Say
Относится к типу: Dialogue Functions | Actor Functions

ScriptEffectElapsedSeconds

Синтаксис:
CODE
ScriptEffectElapsedSeconds

Функция доступна только в скриптах магических эффектов (Magic effect scripts) и возвращает количество секунд, прошедших с последнего обновления магического эффекта. Внутри блока функция ScriptEffectUpdate возвращает положительное вещественное число, например, 1,8 сек.
Напомним, что всего существует 3 типа магических блоков, используемых для создания магических эффектов:
ScriptEffectStart
ScriptEffectFinish
ScriptEffectUpdate
Относится к типу: Magic Functions

SelectPlayerSpell

Синтаксис:
CODE
SelectPlayerSpell SpellID/ScrollID

Функция SelectPlayerSpell используется для того, чтобы заставить персонажа игрока выбрать указанное заклинание (SpellID) или свиток (ScrollID) в качестве активной магии.
Относится к типу: Magic Functions

SendTrespassAlarm

Синтаксис:
CODE
[ActorID.]SendTrespassAlarm CriminalID

Пример:
CODE
SendTrespassAlarm player

Функция SendTrespassAlarm посылает сигнал тревоги о нарушении границ, как если бы кто-то сообщил актеру (ActorID), что персонаж игрока (либо указанный преступник - CriminalID) нарушил границу частных владений.
Относится к типу: Crime Functions

SetActorAlpha

Синтаксис:
CODE
[ActorID.]SetActorAlpha [0.0 - 1.0]
saa [0.0 - 1.0]

Пример:
CODE
SetActorAlpha .5

Функция SetActorAlpha устанавливает значение alpha для актера ActorID (насколько он непрозрачный). Значение "1.0" означает полную непрозрачность, "0.0" - полную прозрачность.
Замечание: С самого начала у игрового движка Gamebryo (положен в основу игр серии TES) существуют проблемы с оверлеями альфа-каналов - этот параметр может порождать странные эффекты при наложении масок прозрачности на текстуры, такие, например, как прозрачные границы вокруг волос (hair meshes).
Относится к типу: Magic Functions | Actor Functions

SetActorFullName

Синтаксис:
CODE
[ActorID.]SetActorFullName "New Name"

Пример:
CODE
SetActorFullName "My Favorite Horse"

Функция SetActorFullName изменяет отображаемое в игре имя актера (ActorID) на приведенный в качестве параметра "New Name" текст. Обратите внимание, что хотя эта функция и вызывается как указатель на конкретного актера (ActorID - копия базового экземпляра, размещенная в игровом мире), будет изменено имя базового экземпляра, поэтому и все остальные копии этого NPC/существа также получат новое имя.
Относится к типу: Miscellaneous Functions | Actor Functions

SetActorRefraction

Синтаксис:
CODE
[ActorID.]SetActorRefraction [0.0 - 10.0]
sar [0.0 - 10.0]

Примеры:
CODE
SetActorRefraction 3; semi-transparent

CODE
SAR 0; opaque

Функция SetActorRefraction устанавливает значение показателя преломления модели актера ActorID (насколько он прозрачный). Визуально это смотрится так, как будто персонаж сделан из воды или льда. Скриншот:

Заметьте, что установленное значение рефракции будет потеряно, когда актер покинет зону высокой обработки (удалится от камеры).

Похоже, что в этой функции есть небольшой сбой. Если вы загрузите сохранение, находясь в режиме "semi-transparent" (полупрозрачный), то таким и останетесь. Это нужно учитывать при создании эффектов заклинаний, потому что вы не сможете помешать игроку использовать быструю загрузку во время действия эффекта. В качестве решения проблемы в скрипт можно вставить такой код:
CODE
if player.IsSpellTarget thespell == 0
player.sar 0
endif

Относится к типу: Magic Functions | Actor Functions

SetActorValue

Синтаксис:
CODE
[ActorID.]SetActorValue StatName value
SetAV StatName value

Пример:
CODE
SetActorValue Strength 60

Player.SetActorValue Alteration 50

Функция SetActorValue позволяет назначить базовое значение (value) какой-либо характеристики (StatName) вызывающего актера (ActorID) (то значение, которое имеет характеристика без дополнительных модификаторов). Измененный параметр будет отображаться в игре синим цветом, как будто он не изменялся ("Unmodified").
Примечание:
Несмотря на то, что функция вызывается на конкретной копии актера (ActorID), она изменяет значение характеристики базового экземпляра (то есть, изменяется характеристика всех копий данного актера в игровом мире). В этом смысле функция ModActorValue модифицирует характеристику только копии (не базового экземпляра) и потому во многих случаях предпочтительнее.
См. также: Stats List, GetActorValue, GetBaseActorValue, ModActorValue, ForceActorValue
Относится к типу: Actor Value Functions | Statistics Functions | Actor Functions

SetAlert

Синтаксис:
CODE
[ActorID.]SetAlert [0/1]

Функция SetAlert устанавливает вызывающего актера (ActorID) в визульное сотояние боевой тревоги (оружие наголо). Отметим, что это чисто анимационное состояние, никак не затрагивающее его реальное состояние и фактический статус боя. (Прим. Garin: надо полагать, происходит простая замена анимации бездействия на анимацию боевой тревоги).
Относится к типу: AI Functions | Actor Functions

SetAllReachable

Синтаксис:
CODE
[ActorID.]SetAllReachable [0/1]

Функция SetAllReachable - одна из 4 функций, которые могут использоваться, чтобы увеличить производительность игры в ситуациях, где много актеров находятся в бою или передвигаются по сложным маршрутам или выполняют сложные действия. Отметьте, что флаг [0/1] сбрасывается в каждом фрейме, поэтому эта функция в выполняемом скрипте должна вызываться на копии актера (ActorID) в загруженной области.
Когда флаг установлен в "1", персонажи не используют сетку путей. Вместо этого они всегда идут к цели напрямик.
См. также: SetSceneIsComplex, SetAllVisible, SetNoAvoidance
Относится к типу: AI Functions

SetAllVisible

Синтаксис:
CODE
[ActorID.]SetAllVisible [0/1]

Функция SetAllVisible - одна из 4 функций, которые могут использоваться, чтобы увеличить производительность игры в ситуациях, где много актеров находятся в бою или передвигаются по сложным маршрутам или выполняют сложные действия. Отметьте, что флаг [0/1] сбрасывается в каждом фрейме, поэтому эта функция в выполняемом скрипте должна вызываться на копии актера (ActorID) в загруженной области.
Если флаг установлен в "1", то полная проверка видимости (LineOfSight check) персонажа не производится (LineOfSight check). Вместо этого, когда нужно определить, видит ли актер (ActorID) что-либо, проверяется только, видит ли он головы других актеров, вместо полной проверки на видимость всех других частей их тел.
См. также: SetSceneIsComplex, SetAllReachable, SetNoAvoidance
Относится к типу: AI Functions

SetAngle

Снтаксис:
CODE
[ObjectID.]SetAngle axis, degrees

Пример:
CODE
SetAngle x, 100

При вызове функции SetAngle угол поворота вызывающего объекта ObjectID становится равным указанному значению угла (degrees) по одной из указанных осей axis (axis = x, y или z).
См. также: GetAngle, GetHeadingAngle
Относится к типу: Movement Functions

SetAtStart

Синтаксис:
CODE
[ObjectID.]SetAtStart

Функция SetAtStart восстанавливает первоначальное положение и угол поворота вызывающего объекта ObjectID.
Относится к типу: Movement Functions

SetBarterGold

Синтаксис:
CODE
[ActorID|Player.]SetBarterGold float

Пример:
CODE
SetBarterGold 200

Функция SetBarterGold устанавливает у вызывающего актера (ActorID) или игрока количество золота для торговли на указанное в виде параметра (float) значение.
См. также: ModBarterGold
Относится к типу: Statistics Functions | Actor Functions

SetCellFullName

Синтаксис:
CODE
SetCellFullName CellID "New Name"

Пример:
CODE
SetCellFullName HouseForSale "My House"

Функция SetCellFullName изменяет отображаемое в игре имя ячейки (CellID) на указанный текст (“New Name”). Это имя будет указываться на дверях, ведущих в ячейку, и на карте.
Относится к типу: Miscellaneous Functions

SetCellOwnership

Синтаксис:
CODE
SetCellOwnership CellID NPC/FactionID (optional)

Примеры:
CODE
SetCellOwnership WeynonPrioryHouse BladesFaction
SetCellOwnership MyImperialCityHouse

При вызове функции SetCellOwnership владельцем указанной в виде параметра ячейки (CellID) становится указанный NPC или фракция (FactionID). Если второй параметр не указан, ячейка будет принадлежать игроку.
Относится к типу: Crime Functions

SetCellPublicFlag

Синтаксис:
CODE
SetCellPublicFlag CellID, Flag
setpublic CellID, Flag

Пример:
CODE
SetCellPublicFlag MyCell, 1

Функция SetCellPublicFlag устанавливает флаг «общественная» для указанной в виде параметра ячейки (CellID) в "1" или в "0". Общественные ячейки всегда доступны (то есть, нельзя попасться на незаконном проникновении).
Относится к типу: Crime Functions

SetClass

Синтаксис:
CODE
[ActorID.]SetClass Classname

Пример:
CODE
SetClass Warrior

Функция SetClass меняет класс вызывающего актера (ActorID) на указанный в виде параметра новый класс (Classname). Автоматический пересчет характеристик будут выполняться только в том случае, если в свойствах актера установлена соответствующая галочка.
Относится к типу: Actor State Functions | Actor Functions

SetCombatStyle

Синтаксис:
CODE
[ActorID.]SetCombatStyle CombatStyleID

Функция SetCombatStyle устанавливает вызывающему актёру (ActorID) новый боевой стиль, указанный в виде параметра CombatStyleID.
Относится к типу: Combat Functions | Actor Functions

SetCrimeGold

Синтаксис:
CODE
[ActorID.]SetCrimeGold float

Пример:
CODE
SetCrimeGold 120.0

Функция SetCrimeGold устанавливает штраф вызывающего актера (ActorID) в указанное в виде параметра float значение.
См. также: GetCrimeGold, ModCrimeGold
Относится к типу: Crime Functions

SetDestroyed

Синтаксис:
CODE
[RefID.]SetDestroyed flag
SetDestroyed 1

Функция SetDestroyed позволяет пометить вызывающую копию объекта (RefID) как "уничтожен", если установить указанный в виде параметра flag в "1". Вызов этой функции с флагом "0" уберет эту отметку. Принимаются только целочисленные значения.
Это может быть полезно для использоваться на двери или активаторе, чтобы показать, что они не функционируют. Разрушенные двери/активаторы не выводят текст при наведении на них курсора и их нельзя активировать. Объекты, относящиеся к уровневым спискам, не будут создавать новыех существ.
Примечания:
Когда функция вызывается на актере, это не даст возможности с ним говорить, ограбить или обворовать его.
Относится к типу: Miscellaneous Functions

SetDoorDefaultOpen

Синтаксис:
CODE
[DoorID.]SetDoorDefaultOpen flag

Функция SetDoorDefaultOpen устанавливает сложность замка вызывающих дверей (DoorID) в значение по умолчанию (см. GetDoorDefaultOpen). Команда имеет смысл только для дверей, которые отмечены в редакторе как постоянные (persistent); непостоянные двери "забудут" все изменения, когда ячейка выгрузится из памяти.
Относится к типу: Miscellaneous Functions

SetEssential

Синтаксис:
CODE
SetEssential ActorBase bool

Пример:
CODE
SetEssential MyFavoriteNPC 1
SetEssential MyLeastFavoriteNPC 0

Функция SetEssential позволяет установить флаг "essential" для указанного в виде параметра ActorBase актера (в "1" или в "0"). Когда этот флаг установлен в "1", NPC (creature) становится неубиваемым - если его здоровье понижается до 0 и ниже, актер просто потеряет на время сознание. Обычно это используется, чтобы предотвратить случайное либо намеренное убийство игроком квестового персонажа, который необходим для успешного завершения определенного квеста.
Примечание: Отметьте, что это может работать как оператор возврата, если в качестве ActorBase указать ref-переменную (указатель на актера).
Относится к типу: Statistics Functions | Actor Functions

SetFactionRank

Синтаксис:
CODE
[ActorID.]SetFactionRank FactionID, newRank

Пример:
CODE
SetFactionRank FightersGuild, 0
CaranyaRef.SetFactionRank MagesGuild, -1

Функция SetFactionRank устанавливает для вызывающего актера (ActorID) новый ранг (newRank) в указанной фракции FactionID. Если актер членом фракции не является, функция включит его в члены фракции в указанном новом ранге.
При вызове функции с параметром newRank = -1 актер из фракции будет исключен:
CODE
SetFactionRank FactionID, -1; это удалит актера из указанной фракции.
Примечания:
Ранг во фракции устанавливается на базовом объекте актера также, как и на его копии. Любой актер, порожденный от этого базового объекта (уровневые существа, функция PlaceAtMe, и так далее), будет иметь эту фракцию и тот же ранг. Это не является проблемой для переименованных персонажей, которые являются уникальными в игровом мире, но может быть проблемой для случайно генерируемых, неименованных персонажей или существ.
См. также: GetFactionRank, GetFactionRankDifference, ModFactionRank
Ссылки: SetFactionRank/Reference
Относится к типу: Faction Functions | Actor Functions

SetFactionReaction

Синтаксис:
CODE
SetFactionReaction FactionID, TargetFactionID, modValue

Пример:
CODE
SetFactionReaction FightersGuild playerFaction -20

Функция SetFactionReaction позволяет установить реакцию (отношение) фракции FactionID ко второй фракции (TargetFactionID) на указанное значение (modValue).
См. также: Category:Factions, ModFactionReaction, GetFactionReaction, GetFactionRank, GetFactionRankDifference, ModFactionRank
В скриптах Обливиона используется в 38 скриптах, в т.ч.:
ArenaAnnouncerScript
ArenaBlueDoorICMonstersScript
ArenaBlueDoorOutsideICScript
Относится к типу: Faction Functions

SetForceRun

Синтаксис:
CODE

[ActorID.]SetForceRun iForceRunFlag

Функция SetForceRun включает/выключает принудительный бег для вызывающего актера (ActorID) с помощью установки флага iForceRunFlag. При установке в "1" актер бежит, при "0" - нормальный режим.
Относится к типу: Actor State Functions

SetForceSneak

Синтаксис:
CODE
[ActorID.]SetForceSneak iForceSneakFlag

Функция SetForceSneak включает или выключает для вызывающего актера (ActorID) режим "красться" с помощью установки флага iForceSneakFlag. При "1" - режим "красться" включен, при "0" - возврат к нормальному режиму.
Относится к типу: Actor State Functions | Actor Functions

SetGhost

Синтаксис:
CODE
[ActorID.]SetGhost iGhostFlag (0 или 1)

Функция SetGhost включает/выключает статус "призрака" для вызывающего актёра (ActorID). Когда он в статусе призрака:
Актёр не вступает в драку.
Актёр не может быть ранен ни оружием, ни заклинанием.
Актёр будет подвержен водным повреждениям и некоторыми ловушками.
Актёр МОЖЕТ наносить повреждения. Так, использование player.SetGhost 1 не будет препятствовать игроку убивать его врагов.
Персонаж может вступать в разговор и с ним можно начать разговор.
Персонаж не может быть обокраден.
См. также: GetIsGhost
Относится к типу: Actor State Functions | Actor Functions

SetIgnoreFriendlyHits

Синтаксис:
CODE
[ActorID.]SetIgnoreFriendlyHits iIgnoreHits

Функция SetIgnoreFriendlyHits используется, чтобы заставить вызывающего актера (ActorID) игнорировать дружественные удары (и запретить ответные). Режим устанавливается с помощью флага iIgnoreHits. При "1" актер игнорирует удары, при "0" переключается в обычный режим. Актёр, игнорирующий дружественные удары, будет вести себя следующим образом:
В бою не будет считать дружественные удары - он никогда не будут проявлять агрессию к игроку, независимо от того, сколько раз он получил удар.
Вне боя принимает нормальное число "дружественных боевых ударов" (3), прежде чем проявит агрессию к игроку.
См. также: GetIgnoreFriendlyHits
Относится к типу: Combat Functions | Actor Functions

SetInCharGen

Синтаксис:
CODE
SetInCharGen flag

Пример:
CODE
SetInCharGen 1

Функция SetInCharGen используется для установки игрока в состояние «генерация персонажа» с помощью флага (flag). В этом состоянии игра будет отслеживать использование игроком Боевых, Магических и Воровских навыков для того, чтобы сгенерировать "класс по умолчанию" (default class).
Смотрите также: GetIsClassDefault.
Относится к типу: Player Functions

SetInvestmentGold

Синтаксис:
CODE

[NpcID.]SetInvestmentGold Amount

Примеры:
CODE
MyFavoriteNPC.SetInvestmentGold 500

Функция SetInvestmentGold позволяет установить с помощью параметра Amount количество золота, инвестированного в NPC. Оно добавляется к доступному для обмена золоту актера при торговле и позволяет игроку совершать более крупные сделки, когда он станет экспертом торговли.
Примечание:
Инвестируемое золото не накапливается. Если вы установите кол-во инвестированного золота в 500, а потом в 750, то NPC получит прибавку в 750, а не 1250.
Параметр Amount должен быть целым положительным числом.
См. также: GetInvestmentGold
Относится к типу: Statistics Functions | Actor Functions

SetItemValue

Синтаксис:
CODE
SetItemValue iSetGoldAmount

Функция SetItemValue устанавливает базовую цену предмета, задаваемого с помощью параметра SetGoldAmount. Отметьте, что при вызове на копии функция изменит базовый объект (что приведёт к изменению стоимости всех остальных копий этого объекта в игровом мире).
Относится к типу: Statistics Functions

SetLevel

Синтаксис:
CODE
[ActorID.]SetLevel iNewLevel, LevelToPCFlag, MinLevel (optional), MaxLevel (optional)

Примеры:
CODE
SetLevel 10; устанавливает абсолютный уровень вызывающего актёра на 10
SetLevel 2, 1; устанавличает уровень вызывающего актёра как PC+2
SetLevel 0, 1, 6, 0; устанавличает уровень актёра как PC+0, с минимальным ограничением 6, но не выше текущего уровня игрока

Описание:
Вызов функции SetLevel позволяет изменить уровень NPC или существа (ActorID) либо установкой абсолютного значения iNewLevel, либо увеличить уровень относительно уровня игрока с помощью флага LevelToPCFlag = 1 на указанную в iNewLevel величину.
Параметры:
"iNewLevel" - новый уровень персонажа, способ его установки зависит от флага LevelToPCFlag.
"LevelToPCFlag" - флаг, устанавливающий способ роста уровня вызывающего актера:
"LevelToPCFlag" == 0 устанавливает прямой рост уровня. При этом LevelActor == "iNewLevel". Если флаг LevelToPCFlag явно не указан,
"LevelToPCFlag" == 1 устанавливает рост уровня актера в зависимости от текущего уровня игрока. При этом LevelActor == LevelPC + "iNewLevel".
"MinLevel" и "MaxLevel" соответствуют установкам Calc Min и Calc Max на закладках NPC и Creature в редакторе и работают только в тех случаях, когда флаг LevelToPCFlag установлен в "1". При "LevelToPCFlag" == 0 параметры "MinLevel" и "MaxLevel" не имеют смысла и могут не указываться.
Вызов SetLevel приведет к немедленному перерасчету статистики актера и его снаряжения (как будто игрок только что поднял уровень).
Примечание: Эту функцию можно вызвать на игроке и изменить его уровень, однако это не изменит его статистику (мир не будет "поднимать уровень" с игроком, пока он не пройдет через загрузочную дверь или не покинет ячейку), поэтому это не рекомендуется. Используйте для этих целей AdvancePCLevel.
См. также: AdvancePCLevel
Относится к типу: Actor State Functions | Actor Functions

SetNoAvoidance

Синтаксис:
CODE
SetNoAvoidance [0/1]

Функция SetNoAvoidance - одна из 4 функций, которые могут использоваться, чтобы увеличить производительность игры в ситуациях, где много актеров находятся в режиме боя или выполняются другие сложные действия. Отметьте, что флаг сбрасывается в каждом фрейме, поэтому эта функция в выполняемом скрипте должна вызываться на копии (объекте) в загруженной области.
Когда флаг установлен, актеры избегают сложных перемещений, например, они перестают открывать двери и обходить друг друга.
См. также: SetSceneIsComplex, SetAllReachable, SetAllVisible
Относится к типу: AI Functions

SetNoRumors

Синтаксис:
CODE
[ActorID.]SetNoRumors [0/1]

Функция SetNoRumors позволяет принудительно установить флаг "No Rumors" для вызывающего NPC (ActorID), который управляет появлением топика "Слухи" в диалоге NPC с игроком.
Относится к типу: Dialogue Functions | Actor Functions

SetOpenState

Синтаксис:
CODE
[DoorID.]SetOpenState flag

Пример:
CODE
SetOpenState 1

Функция SetOpenState позволяет визульно, с соответствующей анимацией, закрыть (flag == "0") или открыть (flag == "1") вызывающие двери (DoorID), если они загружены и находятся в одной ячейке с игроком. Не влияет на открытые по умолчанию двери.
См. также: GetOpenState
Относится к типу: Miscellaneous Functions

SetOwnership

Синтаксис:
CODE
[RefID.]SetOwnership NPC/FactionID (optional)

Пример:
CODE
SetOwnership BladesFaction
RewardSwordRef.SetOwnership

Владельцем копии объекта (RefID), на которой вызывается функция SetOwnership, становится указанный в виде параметра (NPC/FactionID) NPC или фракция. Если второй параметр не указан, владельцем становится игрок.
Относится к типу: Object Functions

SetPCExpelled

Синтаксис:
CODE
[Player.]SetPCExpelled FactionID iFlag

Пример:
CODE
SetPCExpelled MagesGuild 1; изгнать игрока из фракции MagesGuild и установить флаг "expelled" в состояние True (истина) - "изгнан"

Функция SetPCExpelled позволяет установить для персонажа игрока флаг "expelled", устанавливающий признак изгнания его из фракции:

Когда iFlag равен 1, игрок выгоняется из указанной фракции (FactionID) и изменяет "expelled" флаг на «истину» Если игрок находится в ячейке, которой владеет указанная фракция, он автоматически будет телепортирован через ближайшую дверь.
Когда флаг iFlag установлен в 0, флаг "expelled" сбрасывается.
См. также: GetPCExpelled

Относится к типу: Faction Functions | Player Functions

SetPCFactionAttack

Синтаксис:
CODE
SetPCFactionAttack FactionID iFlag

Пример:
CODE
SetPCFactionAttack ThievesGuild 0

Флаг «faction attack» устанавливается каждый раз, когда игрок нападает на члена соответствующей фракции.
Функция SetPCFactionAttack для указанной фракции (FactionID) позволяет установить этот флаг принудительно, используя параметр iFlag == 1. Чтобы очистить его, используйте iFlag == 0.
См. также: GetPCFactionAttack
Относится к типу: Faction Functions | Player Functions

SetPCFactionMurder

Синтаксис:
CODE
SetPCFactionMurder FactionID iFlag

Пример:
CODE
SetPCFactionMurder ThievesGuild 0

Флаг "faction murder" устанавливается каждый раз, когда игрок убивает члена указанной фракции. Функция SetPCFactionMurder позволяет для указанной фракции (FactionID) установить этот флаг принудительно, используя iFlag = 1.
Чтобы очистить функцию, используйте iFlag = 0.
См. также: GetPCFactionMurder
Относится к типу: Faction Functions | Player Functions

SetPCFactionSteal

Синтаксис:
CODE
SetPCFactionSteal FactionID iFlag

Пример:
CODE
SetPCFactionSteal ThievesGuild 0

Флаг "faction steal" устанавливается каждый раз, когда игрок ворует у члена указанной фракции (несмотря на то, было ли это преступлением, или нет).
Функция SetPCFactionSteal позволяет для указанной фракции (FactionID) установить этот флаг принудительно, используя iFlag = 1.
Чтобы очистить функцию, используйте iFlag = 0.
См. также: GetPCFactionSteal
Относится к типу: Faction Functions | Player Functions

SetPCFactionSubmitAuthority

Синтаксис:
CODE
SetPCFactionSubmitAuthority GuardFactionID iFlag

Пример:
CODE
SetPCFactionSubmitAuthority ICFaction 0

Флаг «faction submit authority» устанавливается автоматически каждый раз, когда игрок подчиняется страже (идет в тюрьму или платит штраф) указанной фракции.
Функция SetPCFactionSubmitAuthority позволяет для указанной фракции (FactionID) установить этот флаг принудительно, используя iFlag = 1.
Для очистки функции используйте iFlag=0.
См. также: GetPCFactionSubmitAuthority

Относится к типу: Faction Functions | Player Functions

SetPCFame

Синтаксис:
CODE
SetPCFame value

Пример:
CODE
SetPCFame 52

Функция SetPCFame устанавливает значение «славы» для игрока, указанное в виде параметра "value". Только игрок имеет значения «славы» и «позора».
Относится к типу: Player Functions

SetPCInfamy

Синтаксис:
CODE
SetPCInfamy value

Пример:
CODE
SetPCInfamy 52

Функция SetPCInfamy устанавливает значение «позора» для игрока с помощью параметра "value". Только игрок имеет значения «славы» и «позора».
Относится к типу: Player Functions

SetPCSleepHours

Синтаксис:
CODE
SetPCSleepHours iHours

Пример:
CODE
SetPCSleepHours iHours

Функция SetPCSleepHours заставляет игрока принудительно спать указанное в виде параметра iHours кол-во часов.
Примечание: Эта функция не вызовет автоматическое повышение уровня, даже если у игрока есть нужное кол-во повышений навыка.
См. также: GetPCSleepHours
Относится к типу - Player Functions

SetPackDuration

Синтаксис:
CODE
SetPackDuration iDuration

Примеры:
CODE
SetPackDuration 2.0

Функция SetPackDuration устанавливает продолжительность текущего пакета актера в течение времени iDuration (в минутах игрового времени). Это может быть полезно в ситуациях, где вы хотите, чтобы пакет выполнялся меньше часа.
Относится к типу: AI Functions | Actor Functions

SetPos

Синтаксис:
CODE
[ObjectID.]SetPos axis, pos

Пример:
CODE

SetPos y, 100.56

Функция SetPos изменяет координаты (pos) объекта (ObjectID) в мире по одной из осей (axis) (x, y, z). Координаты имеют тип float.
Относится к типу: Movement Functions

SetQuestObject

Синтаксис:
CODE
SetQuestObject ObjectID flag

Пример:
CODE
SetQuestObject Jauffre 1
SetQuestObject QuestFoozle 0

Функция SetQuestObject устанавливает флаг (flag) "квестовый предмет" на указанном в виде параметра объекте или актере (ObjectID) в "1" или "0". Когда флаг установлен в "1":
NPC/Существо – имеют приоритет в обновлениях, когда игрок изменят состояние процесса. Например, "квестовые" NPC будут следовать за игроком в другую локацию через дверь быстрее. Трупы не будут "подчищаться", когда придет время обновлять ячейку.
Предметы инвентаря – игрок не может продавать или сбрасывать "квестовые предметы".
Относится к типу: Object Functions

SetRestrained

Синтаксис:
CODE
SetRestrained flag

Примеры:
CODE
SetRestrained 1

Функция SetRestrained устанавливает актера в удержанное состояние (или очистить состояние). Удержанные актеры не будут перемещаться от их текущих координат, но будут продолжать "думать" (пакеты выбора, выполнение охранных функций и поднятие тревоги), и идти на диалог.
Примечание: Когда функция вызвана на персонаже игрока, он сможет смотреть вверх или вниз, но вправо или влево нет.
Смотрите также: SetUnconscious
Относится к типу: AI Functions | Actor Functions

SetRigidBodyMass

Синтаксис:
CODE
SetRigidBodyMass fMass

Пример:
CODE
SetRigidBodyMass 80

Функция SetRigidBodyMass устанавливает массу предмета с помощью параметра fMass, которая определяет, насколько далеко он двигается, когда игрок толкает его или ударяет оружием. Допустимые значения 0....100, значение 100 означает, что предмет не двигается.

Относится к типу: Object Functions

SetScale

Синтаксис:
CODE
[ObjectID.]SetScale float

Примеры:
CODE
SetScale 1.5

Функция SetScale устанавливает масштабный множитель размера вызывающего объекта (ObjectID), указанный в виде параметра float. При множителе, равном "1.0", размеры объекта не изменятся, при "2.0" - увеличатся в два раза, и т.д. Масштаб ограничен диапазоном 0.5.....2.0. Функция может принимать значения и вне этого диапазона, но дальнейшего изменения размера не последует.
Примечания:
Размер ограничен значениями от 0.5 до 2.0. Функция может принимать значения и вне этого диапазона, но дальнейшего изменения размера не последует.
Когда функция используется на игроке в режиме от первого лица, изменения видны не будут, если переключиться на вид от третьего лица. Хотя точка обзора игрока в режиме от первого лица изменится (станет выше или ниже), но не настолько, чтобы это было визуально заметно.
Когда функция используется на NPC, параметр функции используется как множитель для базового значения размера персонажа. Например, Высокий Эльф имеет размер 1.1, вызывая на нем SetScale 1.5, мы получим размер 1.65, и именно это значение будет возвращаться функцией GetScale.
См. также: ModScale.
Относится к типу: Statistics Functions

SetSceneIsComplex

Синтаксис:
CODE
SetSceneIsComplex [0/1]

Функция SetSceneIsComplex используется, чтобы упростить обработку сложных комплексных сцен в ситуациях, когда в бой вовлечено множество актеров.
Эту функцию нужно использовать осторожно, и только тогда, когда вы уверены, что контролируете ситуацию, поскольку ее использование существеннно изменяет поведение актеров в игре по сравнению с игровыми установками.
Важное замечание: Флаг обнуляется в каждом фрейме, поэтому для того, чтобы флаг "комплексной сцены" постоянно устанавливался, вам нужно вставить эту функцию в скрипт, который выполняется на объекте в какой-нибудь загруженной области (например, это может быть дверь или активатор).
Важные изменения, которые вызываются установленным в "1" флагом "комплексной сцены":
Уменьшение детализации для анимации и звука.
Проверки отношения к цели - это невыполнение для актеров части проверок "должен атаковать" ("should attack") непосредственно во время боя.
Во время боя не проверяются ограничения на передвижения (актеры могут на объектах отступить в сторону и т.п.)
Низкие и ниже среднего уровни процесса не обновляются - это очень важно! Это означает, что для актеров, которые не находятся в загруженной области, НЕ БУДУТ ОБНОВЛЕНЫ установленные флаги до тех пор, пока они не пройдут через загруженную, например, дверь. И это главная причина, из-за которой эта функция должна использоваться с особой осторожностью.
Появляющиеся в загруженной области трупы убираются чаще.
Нахождение пути отключено (актеры не будут открывать двери или обходить друг друга)
См. также: SetAllReachable, SetAllVisible, SetNoAvoidance
Относится к типу: AI Functions

SetShowQuestItems

Синтаксис:
CODE
SetShowQuestItems flag

Функция SetShowQuestItems используется для того, что показывать или прятать "квестовые" предметы в инвентаре игрока (обычно игрок не может сбрасывать и продавать "квестовые" предметы). Когда флаг flag == 1, "квестовые" предметы будут показаны в инвентаре (по умолчанию). Когда flag = 0, "квестовые" предметы не отображаются.
Очевидно, что эта функция должна использоваться в очень редких случаях, когда нужно, чтобы игрок не потерял навсегда свои "квестовые" предметы.
Относится к типу: Object Functions

SetSize

Синтаксис:
CODE
SetSize iValue

Примеры:
CODE
SetSize 10.0; Увеличит рост человека на 10 ед, т.е. он составит 128+10=138 ед.
SetSize -7.0; Уменьшит рост человека на 7 ед, т.е. он составит 128-7=121 ед.
SetSize 0.0; Возвращает рост человека к стандартному значению 128 ед.

Вызов функции SetSize уменьшает/увеличивает рост человека на указанную величину iValue, являющуюся вещественным числом с плавающей точкой. Стандартный рост человека в империи Тамриэль составляет 128 единиц (units).
Относится к типу: Miscellaneous Functions | Actor Functions

SetStage

Синтаксис:
CODE
SetStage QuestID StageIndex

Пример:
CODE
SetStage MS27 30

Функция SetStage устанавливает указанную в виде параметра стадию (StageIndex) квеста (QuestID) в состояние "Выполнено" (Completed). Результаты выполнения различных элементов стадии квеста могут использоваться в качестве условий (результаты поисков, добавление записей в квестовый журнал, завершение квеста, если это необходимо).
Примечания:
Если это первая стадия квеста, она будет добавлена в журнал и игрок получит на экране сообщение "Квест добавлен" ("Quest Added").
Если квест в настоящее время не выполняется, вызов SetStage автоматически его запустит.
Всегда помните, что Setstage обрабатывает (принимает) только те стадии, которые определены на вкладке "Quest Stages" этого квеста. Если Вы попытаетесь установить стадию квеста, которая не определена, индекс стадии вообще не изменится.
Скрипт в поле "Result" стадии квеста будет выполнен сразу же после вызова функции SetStage. Главный скрипт, содержащий вызываемую функцию SetStage, будет приостановлен до тех пор, пока не выполнится скрипт в поле "Result". Возможно даже использование команды SetStage и в самом поле "Result". В этом случае новый скрипт в этом поле выполнится вместо команды SetStage, а старый скрипт продолжится после того, как закончится новый.
Невозможно определить переменные в скрипте поля "Result" стадии квеста! Этот текст скомпилируется, но переменные будут всегда равны "0"
См. также: GetStage, GetStageDone
Относится к типу: Quest Functions

SetUnconscious

Синтаксис:
CODE
[ActorID.]SetUnconscious flag

Примеры:
CODE
SetUnconscious 1

Функция SetUnconscious устанавливает вызывающего актера (ActorID) в бессознательное состояние (или очищает состояние) при помощи указанного в виде параметра флага (flag) со значением "1". Бессознательные актеры не "думают" вообще. Выполняется только анимация бездействия и физическое реагирование на предметы, ударяющие их.
Примечания:
При вызове функции SetUnconscious на персонаже игрока он сможет смотреть только вверх или вниз, но не вправо или влево.
Установка этого флага на актере приведёт к невозможности его нормальной активации (грабежа, карманной кражи или беседы с ним) до тех пор, пока флаг "бессознательности" не будет сброшен (установлен в "0").
Вызов этой функции на актере, находящегося в данный момент в режиме диалога, не приведет к его немедленному завершению, однако в дальнейшем все диалоги станут невозможны до сброса флага.

Относится к типу: AI Functions | Actor Functions

SetWeather

Синтаксис:
CODE
SetWeather WeatherID, WeatherOverrideFlag (optional)

Примеры:
CODE
SetWeather clear
sw OblivionStormTamriel 1

Функция SetWeather устанавливает погоду к указанному в виде параметра типу погоды WeatherID. Погода начнет переход к новой погоде, время перехода - это установка "Trans Delta" на новом объекте погоды. Отметим, что если погода в настоящий момент изменяется, это остановит ее изменение и начнется переход к новой погоде.
Если необязательный флаг WeatherOverrideFlag установлен в "1", то погода останется в текущем состоянии до тех пор, пока не будет вызвана функция ReleaseWeatherOverride, которая установит нормальный ритм смены погоды.
Примечание: Тип погоды можно сменить с помощью двух функций: ForceWeather и SetWeather. Отличаются они только тем, что в первом случае переход произойдёт мгновенно, а во втором - плавно, в соответствии с настройками объекта погоды.
Список доступных типов погоды WeatherID (Console IDs):
Tamriel
CamoranWeather 000370CE
Clear 00038EEE
Cloudy 00038EFE
DefaultWeather 0000015E
Fog 00038EEF
Overcast 00038EEC
Rain 00038EF2
Snow 00038EED
Thunderstorm 00038EF1
OblivionStormTamriel 000836D5
OblivionStormTamrielMQ16 0006BC8B
Oblivion
tba

См. также: ForceWeather, ReleaseWeatherOverride
Относится к типу: Weather Functions

ShowBirthsignMenu

Синтаксис:
CODE
ShowBirthsignMenu

Функция ShowBirthsignMenu вызывает окно выбора знака зодиака. Предыдущий знак будет заменен.
См. также: ShowRaceMenu, ShowClassMenu
Относится к типу: Player Functions

ShowClassMenu

Синтаксис:
CODE
ShowClassMenu

Функция ShowClassMenu открывает окно выбора класса персонажа игрока.
(Команда сбросит текущие навыки игрока)
См. также: ShowBirthsignMenu, ShowRaceMenu
Относится к типу: Player Functions

ShowDialogSubtitles

Синтаксис:
CODE
ShowDialogSubtitles 0/1

Функция ShowDialogSubtitles глобально включает/выключает субтитры к диалогам (убирает близостную проверку).
Относится к типу: Dialogue Functions

ShowEnchantment

Синтаксис:
CODE
ShowEnchantment

Функция ShowEnchantment вызывает в игре диалоговое окно зачарования.
См. также: ShowSpellMaking
Относится к типу: Magic Functions | Player Functions

ShowMap

Синтаксис:
CODE
ShowMap MapMarkerID, enableFastTravel (optional)

Пример:
CODE
ShowMap SecretDungeonMapMarker; показывает SecretDungeon на карте мира, но переместиться туда с помощью fast travel нельзя.
ShowMap NotSoSecretVillageMapMarker, 1; показывает NotSoSecretVillage на карте мира, и туда можно переместиться с помощью fast travel.

Функция ShowMap добавляет указанный маркер (MapMarkerID) на мировую карту игрока. Игрок не сможет переместиться туда с помощью fast-travel, если флаг enableFastTravel не установлен в "1" (равен 0).
Относится к типу: Miscellaneous Functions

ShowRaceMenu

Синтаксис:
CODE
ShowRaceMenu

Функция ShowRaceMenu вызывает окно выбора расы (специальные способности новой расы заменят старые).
См. также: ShowBirthsignMenu, ShowClassMenu
Относится к типу: Player Functions

ShowSpellMaking

Синтаксис:
CODE
ShowSpellMaking

Функция ShowSpellMaking вызывает диалоговое окно создания заклинаний.
См. также: ShowEnchantment
Относится к типу: Magic Functions | Player Functions

SkipAnim

Синтаксис:
CODE
[ActorID.]SkipAnim

Вызов функции SkipAnim приводит к немедленному завершению проигрываемой анимации в текущем фрейме (кадре) для вызывающего актера ActorID.
Примечания:
Действие функции SkipAnim на объект будет продолжаться до тех пор, пока объект остается загруженным в текущей ячейке или не будет вызвано новое проигрывание анимации с помощью функций LoopGroup или PlayGroup.
Функция работает ненадежно и может вызывать мерцание вызывающего актера, а то и полное исчезновение, или же кажущееся отображение в другом месте, чем это есть фактически.
См. также: LoopGroup, PlayGroup
Относится к типу: Animation Functions

StartCombat

Синтаксис:
CODE
[ActorID.]StartCombat TargetActorID

Пример:
CODE
StartCombat SuperGuard

Функция StartCombat принуждает вызывающего актёра (ActorID) начать бой с указанным в виде параметра TargetActorID вторым актером. Вызывающий актёр будет всегда держать целевого актёра первым в своем списке целей (target list).
Относится к типу: Actor Functions | Combat Functions

StartConversation

Синтаксис:

CODE
[ActorID.]StartConversation TargetActorID, TopicID

Примеры:
CODE
StartConversation Joe, SecretTopic
StartConversation player, ForceGreetTopic

Функция StartConversation используется для принудительного начала диалога между двумя NPC на определённую тему. Когда вы вызываете эту функцию на актёре (ActorID), он начинает искать указанного в виде параметра целевого актёра TargetActorID, чтобы начать разговор (TopicID), независимо от того где эти два NPC находятся в настоящее время.
Вы можете также использовать это, чтобы NPC принудительно обратился к игроку, устанавливая игрока как TargetActor. Указанная тема будет использоваться вместо нормального приветственного топика, таким образом Вы можете заставить NPC начать диалог с любой темы, какую захотите (Используйте Greeting, если Вы хотите просто обычное приветствие).
Относится к типу: Dialogue Functions | Actor Functions

StartQuest

Синтаксис:
CODE
StartQuest QuestID

Пример:
CODE
StartQuest SQ09

Функция StartQuest запускает указанный в виде параметра QuestID квест. Это означает, что:
Все вложенные в квест диалоги будет включёны.
Скрипты квеста, если таковые есть, начнут выполняться.
См. также: StopQuest
Относится к типу: Quest Functions

StopCombat

Синтаксис:
CODE
[ActorID.]StopCombat

Функция StopCombat останавливает бой для вызывающего актера ActorID. Однако, если актер все еще агрессивен и имеет низкое расположение к любому из рядом находящихся актеров, он может немедленно возвратиться в бой. Наиболее полезна эта функция в тех случаях, когда требуется остановить бой, который был начат искусственно, с помощью функции StartCombat.
См. также: StartCombat
Относится к типу: Combat Functions | Actor Functions

StopCombatAlarmOnActor

Синтаксис:
CODE
[ActorID.]StopCombatAlarmOnActor
SCAOnActor

Функция StopCombatAlarmOnActor останавливает все атаки и тревоги (alarms) против вызывающего актёра (ActorID).
Относится к типу: Combat Functions | Actor Functions

StopLook

Синтаксис:
CODE
[ActorID.]StopLook

Функция StopLook отменяет вынужденное осматривание для вызывающего актера (ActorID), вызванное функцией Look. Возвращается обычная линия поведения актера.
Смотрите также: Look
Относится к типу: AI Functions | Actor Functions

StopMagicEffectVisuals

Синтаксис:
CODE
StopMagicEffectVisuals MagicEffectID

Пример:
CODE
StopMagicEffectVisuals SABS
sme SHLD

Функция StopMagicEffectVisuals прекращает визуальное проигрывание указанного в виде параметра магического эффекта MagicEffectID.
Смотрите также: PlayMagicEffectVisuals
Относится к типу: Magic Functions

StopMagicShaderVisuals

Синтаксис:
CODE
StopMagicShaderVisuals EffectShaderID

Пример:
CODE
StopMagicShaderVisuals GhostEffect
sms effectShockShield

Функция StopMagicShaderVisuals прекращает визуальное проигрывание указанного в виде параметра EffectShaderID шейдерного эффекта.
Отметьте, что останавливать шейдер, у которого вы указали продолжительность, не обязательно.
Список шейдеров магических эффектов
http://cs.elderscrolls.com/constwiki/index..._Effect_Shaders
Список предустановленных шейдерных эффектов, которые могут применятся в магических эффектах или могут быть проиграны с помощью PlayMagicShaderVisuals:
• creatureAnvilMGPetImp
• creatureEffectGhostAncient
• creatureEffectLichNether
• creatureEffectSkeletonChampion
• creatureEffectSkeletonChampion2
• creatureEffectSkeletonGuardian
• creatureEffectSkeletonHero
• creatureEffectUndeadBlade
• creatureEffectWraithFaded
• creatureEffectWraithGreater
• creatureEffectZombieDread
• effectAbsorb
• effectAtronachFlame
• effectAtronachFrost
• effectAtronachStorm
• effectBurden
• effectCalm
• effectCharm
• effectCommand
• effectDamage
• effectDemoralize
• effectDestruction
• effectDetectLife
• effectDisease
• effectDisintegrateArmor
• effectDrain
• effectEnchantAlteration
• effectEnchantConjuration
• effectEnchantDestruction
• effectEnchantFireDamage
• effectEnchantFrostDamage
• effectEnchantIllusion
• effectEnchantMysticism
• effectEnchantPoison
• effectEnchantRestoration
• effectEnchantShockDamage

• effectEnchantTurnUndead
• effectFireDamage
• effectFireDragon
• effectFireShield
• effectFortify
• effectFortifyFatigue
• effectFortifyHealth
• effectFortifyMagicka
• effectFrenzy
• effectFrostDamage
• effectFrostShield
• effectLock
• effectOpen
• effectParalyze
• effectPoison
• effectRally
• effectReanimate
• effectReflect
• effectReflectDamage
• effectReflectSpell
• effectResistNormalWeapons
• effectRestore
• effectRestoreHealth
• effectShield
• effectShockDamage
• effectShockShield
• effectSiegeCrawler
• effectSilence
• effectSoulTrap
• effectSpellAbsorption
• effectStone
• effectSummonMythicDawn
• effectSunDamage
• effectTelekinesis
• effectTG11Stone
• effectTurnUndead
• effectWeakness
• GhostEffect
• LifeDetected
• testfireshader
• testfrostshader
• testwatershader
Смотрите также: PlayMagicShaderVisuals, List of Effect Shaders
Относится к типу: Magic Functions

StopQuest

Синтаксис:
CODE
StopQuest QuestID

Пример:
CODE
StopQuest SQ09

Функция StopQuest останавливает указанный в виде параметра QuestID квест.
Это означает, что:
Все прикреплённые диалоги будут отключены.
Квестовые скрипты, если они есть, будет остановлены.
Примечание: Эта функция не работает как функция Возврат, когда вызывается из собственного скрипта. Скрипт будет продолжать работать, пока не достигнет последнего блока.
См. также: StartQuest
Относится к типу: Quest Functions

StopWaiting

Синтаксис:
CODE

[ActorID.]StopWaiting PackageID

Функция StopWaiting сообщает вызывающему актеру ActorID, что нужно возобновить нормальное поведение AI-пакета (PackageID) после нахождения в состоянии ожидания (Wait), вызванной командой Wait.
Смотрите также: Wait
Относится к типу: AI Functions | Actor Functions

StreamMusic

Синтаксис:
CODE
StreamMusic "<Filename>"; проиграть указанный файл (кавычки обязательны!)
StreamMusic Public ; играть случайный файл из Data\Music\Public
StreamMusic Explore ; играть случайный файл из Data\Music\Explore
StreamMusic Dungeon ; играть случайный файл из Data\Music\Dungeon
StreamMusic Random ; играть случайный файл из трех папок, указанных выше

Примеры:
CODE
StreamMusic "data\music\special\success.mp3"
StreamMusic "LauncherMusic.wav"
StreamMusic "..\Morrowind\Data Files\Music\Special\morrowind title.mp3"
StreamMusic "C:\WINDOWS\Media\tada.wav"

Функция StreamMusic воспроизводит указанный (в кавычках) звуковой файл.
Примечания:
Bethesda не документировала эту функцию, и она частично сломана/незакончена, так что будьте осторожны и следите за возможным странным поведением.
После завершения проигрывания будет выбран новый файл в соответствии с текущей локацией.
Если вы укажете имя файла, который не существует, то будет проигрываться музыка, звучавшая до этого.
StreamMusic работает в ячейках, где тип музыки установлен как "Dungeon" или "Public". Когда функция используется в ячейке "Default", она лишь заново проигрывает текущий файл.
Чтобы использовать эту команду в ячейке "Default", необходимо сначала вызвать StreamMusic random, затем подождать один фрейм, чтобы команда выполнилась, и затем вызвать Streammusic "<filename>".
StreamMusic не прекращает проигрывание боевой музыки, боевая музыка при вызове функции начнет проигрываться заново.
Путь к имени файла отсчитывается от папки Oblivion, а не Data\Music.
Тип файл не обязательно должен быть MP3. Любой формат, который понимает Windows Media Player, может использоваться в Oblivion (например файлы .MID).
Тип файла даже не обязательно должен быть музыкой; если вы таким образом запустите видео, Oblivion свернется и видеоролик будет показан в дополнительном окне. Когда ролик закончится, Oblivion возобновит свою работу. (Это может привести к нестабильности игры, так что используйте это только для пасхалок.)
Смотрите также: PlaySound, PlaySound3D
Относится к типу: Miscellaneous Functions

T

This

Синтаксис:
CODE
set refVar to This
set refVar to GetSelf

Пример:
CODE
if This == GetActionRef
CODE
set MyQuest.targetRef to This

Функция This является псевдонимом функции GetSelf. Возвращает ссылку на вызвавший объект. Полезна для использования в качестве условий или для установки переменных в других скриптах.
Примечания:
Когда функция вызвана на carriable объекте (оружие или предмет из категории "разное"), возвращённая ссылка имеет допустимое значение только в тех случаях, если объект остаётся в игровом мире.
Эта функция ненадёжна при вызове на игроке (Unreliability with Player).
Функция возвращает "0" при вызове на источнике света.
См. также: Unreliability with Player, GetSelf
Относится к типу: Reference Variable Functions

ToggleActorsAI

Синтаксис:
CODE
[ActorID.]ToggleActorsAI

Функция ToggleActorsAI включает/отключает для вызывающего актера (ActorID) искусственный интеллект AI, анимацию и физику.
См. также: IsActorsAIOff
Относится к типу: Console Functions | Actor Functions

TrapUpdate

Синтаксис:
CODE
TrapUpdate

TrapUpdate - особая функция, позволяющая ловушкам наносить повреждения.
Для того, чтобы ловушка нанесла повреждения, в скрипте должны быть определены следующие переменные:
.
fTrapDamage
Тип переменной - float
Количество повреждений, наносимое ловушкой.
fTrapPushBack
Тип переменной - float
Величина силы отдачи, применяемой к актеру. Должна быть в пределах 0 до 1000.
fTrapMinVelocity
Тип переменной - float
Минимальная скорость, с которой ловушка должна двигаться по отношению к актеру, чтобы нанести повреждение. Значение должно быть BSUnits (128 = 6 футов).
bTrapContinuous
Тип переменной - float
0 = Наносить повреждение при первом контакте. 1 = Постоянно наносить повреждения, пока актер контактирует с ловушкой.
От этого зависит то, как trapdamage и trappushback влияют на актера. Для постоянных ловушек, таких как облака дыма или огонь, повреждения и отдача должны быть малы. Непостоянные ловушки наносят повреждения при первом контакте и не наносят повреждений, пока контакт не прерван и снова не восстановлен.
TrapUpdate можно поместить в блок GameMode; каждый раз, когда она будет вызываться, будет производиться проверка, является ли кто-то в контакте с ловушкой, и будут наносится повреждения, определенные вышеприведенными переменными.
Относится к типу: Miscellaneous Functions

TriggerHitShader

Синтаксис:
CODE
TriggerHitShader iStrength

Функция TriggerHitShader используется для того, чтобы вручную активировать «шейдер удара»("hit shader"), то есть эффект размывания экрана (обычно используется, когда игрок шатается). Часто используется в ловушках, и других активаторах, «сотрясающих землю».
Параметр силы (iStrength) может быть как угодно велик, но значения выше 4 или 5 не приводят к желаемому эффекту.
Относится к типу: Miscellaneous Functions

U

UnequipItem

Синтаксис:
CODE
UnequipItem ObjectID NoEquipFlag

Примеры:
CODE
UnequipItem FavoriteCuirass
CODE
player.UnequipItem AmuletOfKings 1

Функция UnequipItem принуждает игрока снять объект (ObjectID). Если NoEquipFlag = 1, актеры (включая игрока) не смогут надеть предмет.
См. также: EquipItem
Относится к типу: Object Functions | Actor Functions

Unlock

Синтаксис:
CODE
Unlock unlockAsOwner (не обязательно)

Примеры:
CODE
Unlock
Unlock 1

Функция Unlock открывает (убирает замок) с двери или контейнера. Если вы используете необязательный флаг UnlockAsOwner == "1", будет считаться, что дверь открыл владелец. Это нужно для целей AI, чтобы определить, является ли попытка входа в ячейку за дверью нарушением частных владений или нет.
Если флаг unlockAsOwner указан явно и равен "0", становится все равно, была ли ячейка помечена как публичная (public) или как частная собственность (private).
См. также: GetLockLevel, GetLocked, Lock
Относится к типу: Miscellaneous Functions

V

VampireFeed

Синтаксис:
CODE
[VampireID.]VampireFeed TargetActorID

Пример:
CODE
VampireFeed CattleNPCRef

Функция VampireFeed добавляет вызывающему вампиру (VampireID) пакет питания на указанного в виде параметра TargetActorID актера.
VampireFeed берет один параметр актора, который становится целью его пакета питания. Эта функция работает только в тех случаях, если вызывающий NPC является вампиром, а цель спит в кровати.
См. также: HasVampireFed
Относится к типу: Actor State Functions | Actor Functions

W

Wait

Синтаксис:
CODE
[ActorID.]Wait PackageID

Функция Wait сообщает вызывающему актеру (ActorID), что нужно перейти в состояние ожидания "wait" для указанного пакета PackageID.
Выполняется только тогда, когда PackageID является текущим пакетом актера, и если текущий пакет - это пакет следования (Follow) или сопровождения (Accompany).
Смотрите также: StopWaiting
Относится к типу: AI Functions | Actor Functions

WakeUpPC

Синтаксис:
CODE
WakeUpPC

Функция WakeUpPC принудительно закрывает меню ожидания/сна, как будто-бы игрок нажал на клавишу Cancel ("отменить").
Относится к типу: Player Functions

WhichServiceMenu

Синтаксис:
CODE
WhichServiceMenu

Функция WhichServiceMenu возвращает значение, соответствующее типу отображаемого в данный момент служебного меню:
CODE
0 = Barter (Бартер)
1 = Repair (Ремонт)
2 = Training (Тренировка)
3 = Recharge (Перезарядка)
4 = Spell Buying (Покупка заклинания)
ПРИМЕЧАНИЕ: Это работает только как функция условия.
Относится к типу: Condition Functions

Y

Yield

Синтаксис:
CODE
[ActorID.]Yield

Функция Yield заставляет вызывающего актера ActorID потерять основную цель. Все остальные цели, если они есть, удаляются и персонаж выходит из боя.
Примечание: Если другой персонаж ударит актера, на котором вызвана yield, бой может начаться снова.
Относится к типу: Combat Functions | Actor Functions

7. Расширитель функций скриптового языка (OBSE v0009a)
Oblivion Script Extender v0009a
Авторы: Ian Patterson (ianpatt) и Stephen Abel (behippo)

Ссылки:
Сайт разработчиков:
http://obse.silverlock.org/
Скачать "Oblivion Script Extender v0009a": http://obse.silverlock.org/download/obse_0009a.zip
Документация (на английском):
http://obse.silverlock.org/obse_command_doc.html
Демонстрационное видео (15MB):
http://obse.silverlock.org/download/OBSE_v0009_Demo.wmv
Официальный форум:
Oblivion Script Extender v0009
Oblivion Script Extender v0009a (Thread 2), inclues info on v0010

7.1 Основные сведения
Oblivion Script Extender представляет собой расширитель функций скриптового языка Обливиона и выполняется после него в теневом режиме. OBSE добавляет новые интересные функции, которые можно с успехом использовать в своих модах. OBSE революционизирует написание скриптов в связи с множеством различных приятных дополнений. Например, в OBSE включены такие очень полезные функции, как IsKeyPressed, GetCurrentSpell и многие другие.

Начал разработку OBSE Ian Patterson. Позднее к нему подключился Stephen Abel. OBSE развивается весьма динамично. В скором времени ожидается выход 10-й версии.

OBSE не делает никаких модификаций в oblivion.exe или TESConstructionSet.exe или в любых других файлах вашей установленной игры, так что не стоит беспокоиться о возникновении побочных эффектов. Отметьте, что расширитель начал разрабатываться уже после выхода последней официальной версии патча 1.1.0.511 к игре TES 4 Oblivion и работает только с ним. Впрочем, в локализованной версии от 1С этот патч уже включен по умолчанию, так что счастливым владельцам официальной версии не стоит беспокоиться. А вот проблемы могут возникнуть, если у вас нелицензионная версия.

Ian Patterson предупреждает, что в случаях, если:
"...в obse_loader.log сообщается, что контрольная сумма не совпадает, то:
Вы можете иметь версию Обливиона, которая не поддерживается. Я использовал английский официальный патч v1.1.0.511, а вот локализованные версии с различными executables или другими патчами работать не будут. Если будет иметься достаточно большой спрос для какой-нибудь подобной версии, я могу в будущем добавить поддержку расширителя и для нее.

Ваш установленный Обливион может быть запорчен. Хакерские или no-cd-патчи могут также изменить контрольную сумму игры, делая невозможным обнаружение установленной версии."
Это означает, что вы должны использовать только официальные версии игры.
Это же относится и к конструктору TES 4 Oblivion. Следует отметить, что существующие русификаторы конструктора также изменяют контрольную сумму, однако есть и хорошие новости - стараниями Boblen`а Ян Паттерсон включил поддержку контрольной суммы русификатора для TES 4 CS от Serj777, за что всем участвовавшим в этом процессе большое спасибо.

7.2 Изменения в новой версии:
Редакция OBSE v0009:
1. Все индивидуальные функции, возвращающие значения (типа GetXXXValue), и известные из более ранних редакций, теперь нормально функционируют.
2. Вводит много устанавливающих (Set) и модифицирующих значения (Mod Value) функций для предметов инвентаря и магических предметов.
3. Включены бета-версии функций управления потоками. Функции Label (метки) и Goto (перейти к) обеспечивают основные функции для организации циклов. Эти функции доступны также под другими именами - как SaveID и RestoreIP.
4. В новой версии:
всего 114 функций: 70 из них обеспечивают целиком новую функциональность;
исправлен дефект, образовывающий циклы при работе с магическими эффектами;
исправлен дефект с ядами при использовании GETMIV - теперь вы можете быть уверены, что возвращаемое значение о ядовитости или иного объекта корректно;
исправлены многочисленные баги и другие замечания за трехмесячный период между между релизами 8-й и 9-й версий OBSE;
включена новая документация в формате HTML.
Редакция OBSE v0009a:
исправлены замеченные баги;
исправлены функции управления потоками;
исправлен возможный баг с функцией SetModel.

7.3 Установка OBSE:
Скачайте Oblivion Script Extender v0009a
Скопируйте файлы obse.dll, obse_editor.dll и obse_loader.exe в вашу папку с Oblivion.
Скопируйте scripttest.esp в вашу папку с плагинами (Oblivion plugins) и включите его в лаунчере.
Запустите Oblivion, используя файл obse_loader.exe из папки Oblivion.
Чтобы создавать свои моды с использованием OBSE, вам нужно запустить obse_loader.exe с флагом -editor. Это может быть также сделано в командной строке (обычно, Start -> Programs -> Accessories -> Command Prompt) или же с помощью создания специального ярлыка, который запустит файл obse_loader.exe с дополнительными параметрами, например, вот так:
CODE
"...wherever it's installed\Oblivion\obse_loader.exe" -editor

7.4 Типы скриптовых функций OBSE v0009a
Все скриптовые функции OBSE v0009a разбиты на 13 категорий:
OBSE Console Functions - консольные функции;
OBSE Debug Functions - отладочные функции;
OBSE Deprecated Functions - устаревшие функции;
OBSE Hero Functions - функции главного героя;
OBSE Input Functions - функции ввода;
OBSE Inventory Functions - функции для работы с инвентарем;
OBSE Item Functions - функции для работы с предметами;
OBSE Magic Functions - магические функции;
OBSE Math Functions - математические функции;
OBSE Miscellaneous Functions - функции различного назначения (прочие);
OBSE Actor Functions - функции для работы с актерами;
OBSE Flow Control Functions - функции управления потоками;
OBSE Reference Functions - функции для работы с ref-переменными (ссылками)

Эти функции, которые ранее были доступны только в консоли, теперь могут использоваться также и в скриптах, но их корректная работа не гарантируется. Главным образом, это может касаться их несохранения в игровых Save-файлах.
Con GetINISetting
Con HairTint
Con ModWaterShader
PurgeCellBuffers
Con RefreshINI
Con RunMemoryPass
Con SetCameraFOV
Con SetClipDist
Con SetFog
Con SetGameSetting
SetNumericGameSetting

Con SetGamma
Con SetHDRParam
Con SetImageSpaceGlow
Con SetINISetting
Con SetSkyParam
Con SetTargetRefraction
Con SetTargetRefractionFire
Con SexChange
Con ToggleDetection
Con WaterDeepColor
Con WaterReflectionColor
Con WaterShallowColor

Эти функции могут быть полезны модостроителям для отладки своих собственных модов.
GetOBSEVersion
PrintToConsole
OBSE Deprecated Functions:
Функции, которые больше не используются в новых версиях OBSE, но все еще работающие для обеспечения совместимости.
GetActiveSpell был переименованный GetPlayerSpell.
SetActiveSpell был исключен благодаря наличию SelectPlayerSpell в Oblivione.
GetInventoryItemType - это переименованный GetInventoryObject.
а GetEquipmentSlotType -переименованный GetEquippedObject, потому что эти новые имена более согласованы с модостроительным жаргоном Обливиона и интуитивно понятны.
GetActiveSpell
GetEquipmentSlotType
GetInventoryItemType
SetActiveSpell

Функции, работающие только с главным героем, персонажем игрока.
В этой категории только одна функция:
GetPlayerSpell

Эти функции либо возвращают информацию о нажатии игроком клавиш клвиатуры (т.е. читает keypresses), или либо сообщает игре, что игрок является inputting некоторой информации (т.е. симулирует нажатие на клавишу).
AHammerKey
DisableKey
DisableMouse
EnableKey
EnableMouse
GetAltControl
GetControl
GetKeyPress
GetNumKeysPressed
HammerKey
HoldKey
IsKeyPressed
IsKeyPressed2
MoveMouseX
MoveMouseY
ReleaseKey
SetMouseSpeedX
SetMouseSpeedY
TapKey
UnhammerKey

Функции этой категории OBSE обеспечивают скрипты более подробной информацией о предметах, содержащихся в инвентаре актеров и контейнерах.
GetEquippedObject
GetInventoryObject
GetNumItems

Функции этой категории OBSE обеспечивают скрипты более подробной информацией о предметах.
CloneForm
GetArmorAR
GetBaseObject
GetCurrentValue
GetEquippedCurrentValue
GetEquippedObjectValue

GetObjectType
GetObjectValue
IsActivator
IsAlchemyItem
IsAmmo
IsArmor
IsBook
IsClothing
IsContainer
IsDoor
IsFurniture
IsIngredient
IsKey
IsLight
IsSoulGem
IsWeapon
ModActorValue2
SetModelPath

Функции OBSE, работающие с магией и заклинаниями.
GetPlayerSpell

Эти функции OBSE позволят использовать новые расширенные математические операции в скриптах создаваемых вами модов.
Реализовать подобные возможности можно и без использования функций OBSE. Для этого можно применить длинные формулы, о которых можно прочитать здесь.
Abs
Acos
Asin
Atan
Atan2
Ceil
Cos
Cosh
Exp
Floor
Log
Log10
Pow
Rand
Sin
Sinh
SquareRoot
Tan
Tanh

Функции OBSE, которые сложно отнести к какой-либо другой категории.
GetGameLoaded
GetParentCell

Функции для работы с актерами, к которым можно отнести существа (Creatures) и персонажи (NPC`s).
GetActorLightAmount
IsRefEssential
SetRefEssential

Функции, которые обнаруживают состояние программы или изменяют течение выполнения скриптов.
Примечания:
Циклические функции действия подобно GoTo (идти к) и Label (метка) в оригинальной редакции OBSE v0009 содержали небольшие баги. Убедитесь,
вы загрузили более позднюю версию OBSE v0009a перед их использованием.
Отметьте, что некоторые функции, возможно, ведут себя непредвиденно, когда вызываются внутри цикла. Пример - функция Drop (бросить предмет): пока цикл выполняется непосредственно, то все нормально, но при последующих вызовах функции окажется, что они будут обрабатываться игровым движком последовательно.
Это означает, что если вы бросаете предметы в разные отрезки времени, но в пределах незаконченного цикла, то будут браться только отдельные фреймы, прежде, чем все предметы окажутся упавшими.
Выйдите из текущей ячейки прежде, чем все предметы окажутся на земле...
GetGameLoaded
GoTo

Label
RestoreIP
SaveIP

Функции OBSE, работающие с копиями объектов и ref-переменными.
GetActorLightAmount
GetBaseObject
GetCurrentValue
GetEquippedCurrentValue
GetEquippedObject
GetEquippedObjectValue
GetInventoryObject
GetNumItems
GetObjectType
GetObjectValue
GetParentCell
Con HairTint
IsActivator
IsAlchemyItem
IsAmmo
IsArmor
IsBook
IsClothing
IsContainer
IsDoor
IsFurniture
IsIngredient
IsKey
IsLight
IsRefEssential
IsSoulGem
IsWeapon
ModActorValue2
SetModelPath
SetRefEssential
Con SexChange

7.5 Официальная документация по командам OBSE v0009a
Перевод: Boblen
OBSE @ Oblivion Construction Set Wiki

Содержание

Синтаксис функций

(returnValueName: returnValueType) reference.FunctionName parameter1:type parameter2:type
(ВозвращаемоеЗначениеНазвание: ВозвращаемоеЗначениеТип) Объект.НазваниеФункции параметр1:тип параметр2:тип

optional – опциональные параметры или ссылки (references) набраны курсивом

Тип параметра, указываемого после двоеточия:
float - вещественное положительные или отрицательные числа
long - положительное целое большое число
short - положительное целое число, меньшее long
ref - (Reference) ссылка на объект (обычно objectID)

chars - 4-х символьный код магического эффекта (например: FIDG, Z001)
string - (строка) набор символов, заключенный в кавычки

Правила вызова функций:
По ссылке (Reference) или ObjectID
CODE
reference.FunctionName someParameters objectID:ref

Примеры:
CODE
(health:long) reference.GetObjectHealth objectID:ref
(oldEnchantment:ref) reference.SetEnchantment nuEnchantment:ref objectID:ref

Эти функции могут быть вызваны на объекте или получить objectID в качестве аргумента. Если передан objectID, он получает преимущество над вызывающим объектом.
Только по ссылке (Reference):
CODE
Reference.FunctionName someParameters

Примеры:
CODE
(oldPoison:ref) reference.SetEquippedWeaponPoison nuPoison:ref

Эти функции должны быть вызваны на объекте.
Только по ObjectID:
CODE
FunctionName someParameters objectID:ref
(masterLevel:short) GetSpellMasteryLevel spell:ref
(effectCode:long) GetNthEffectItemCode magicItem:ref whichEffect:long

Свойства

Свойства – это наборы родственных значений и функций, которые применяются к различным типам объектов Обливиона. Для целей документации я собрал родственные значения вместе. Любой тип, в котором есть указанное свойство, содержит все перечисленные значения и все функции свойства могут быть применены к типу.
Attacking (Атакующие)
Breakable (Разрушаемые)
Class (Класс)
Container (Контейнер)
Edible
Enchantable
Equippable
Inventory (Инвентарь)
Magic (Магия)
Named (Именованные)
Simple
Wearable
Attacking (Атакующие)
Типы с этим свойством используются для атак.
Значения:
Attack Damage - long - базовое значение повреждения от атаки; используется в формуле с соответствующим навыком, чтобы рассчитать реальное повреждение от атаки
Speed - float - фактор скорости; используется в формуле, чтобы определить коэффициент атаки
Ignores Resistance flag - bool - определяет, будет ли оружие игнорировать сопротивление обычному оружию.
Функции:
GetAttackDamage – возвращает базовое повреждение от атаки
CODE
(damage:long) reference.GetAttackDamage objectID:ref
SetAttackDamage – задает базовое повреждение от атаки
CODE
(nothing) reference.SetAttackDamage nuDamage:long objectID:ref
ModAttackDamage – изменяет базовое повреждение от атаки вверх или вниз
CODE
(nothing) reference.ModAttackDamage modifyBy:float objectID:ref
GetWeaponSpeed – возвращает скорость оружия
CODE
(speed:float) reference.GetWeaponSpeed objectID:ref
SetWeaponSpeed – задает скорость оружия

CODE
(nothing) reference.SetWeaponSpeed nuSpeed:float obejctID:ref
ModWeaponSpeed – изменяет скорость оружия вверх или вниз
CODE
(nothing) reference.ModWeaponSpeed modifyBy:float obejctID:ref
GetIgnoresResistance – возвращает, установлен ли флаг ignores normal damage resistance
CODE
(ignores:bool) reference.GetIgnoresResistance objectID:ref
SetIgnoresResistance – устанавливает флаг ignores normal damage resistance
CODE
(nothing) reference.SetIgnoresResistance shouldIgnore:bool objectID:ref
Breakable (Разрушаемые)
Типы с этим свойством могут быть разрушены.
Значения:
Health - long - базовое здоровье объекта. Не может быть отрицательным.
Current Health - float - текущее здоровье объекта. Объект поврежден, когда текущее значение меньше базового, и разрушен, когда текущее здоровье достигает 0. Эффективность объекта может уменьшаться с повышением повреждений.
Функции:
GetObjectHealth – возвращает базовое здоровье объекта
CODE
(health:long) reference.GetObjectHealth objectID:ref
SetObjectHealth – задает базовое здоровье объекта
CODE
(nothing) reference.SetObjectHealth nuHealth:long objectID:ref
ModObjectHealth – изменяет базовое здоровье объекта вверх или вниз
CODE
(nothing) reference.ModObjectHealth modifyBy:float objectID:ref
GetEquippedCurrentHealth – получает текущее здоровье объекта в указанном слоте экипировки
CODE
(health:long) reference.GetEquippedCurrentHealth slot:short
SetEquippedCurrentHealth – задает текущее здоровье объекта в указанном слоте экипировки
CODE
(nothing) reference.SetEquippedCurrentHealth nuHealth:long slot:short
ModEquippedCurrentHealth – изменяет текущее здоровье объекта в указанном слоте экипировки вверх или вниз
CODE
(nothing) reference.ModEquippedCurrentHealth modifyBy:float slot:short
Class (Класс)
Типы с этим свойством содержат класс игрока.
Значения:
Attributes - short - атрибута класса
Skills - short - навыки класса
Specialization - short - специализация класса
Функции:
GetClass – возвращает класс NPC
CODE
(class:ref) reference.GetClass
GetClassAttribute – возвращает N-ый атрибут класса
CODE
(skill:short) GetClassAttribute whichAttribute:short class:ref
GetClassSkill – возвращает N-ый навык класса
CODE
(attribute:short) GetClassSkill whichSkill:short class:ref
GetClassSpecialization – возвращает специализацию класса
CODE
(specialization:short) GetClassSpecialization class:ref
IsClassSkill – возвращает, является ли переданный навык навыком класса
CODE
(isClassSkill:bool) IsClassSkill skill:short class:ref
IsMajor – возвращает, является ли переданный навык главным навыком класса
CODE
(isClassSkill:bool) IsMajor skill:short class:ref
Container (Контейнеры)
Любые объекты, имеющие инвентарь.
Функции:
GetNumItems – возвращает число различных типов объектов в контейнере
CODE
(count:long) reference.GetNumItems
GetInventoryObject – возвращает objectID N-ого предмета в контейнере

CODE
(objectID:ref) reference.GetInventoryObject whichObject:long
IsContainer – возвращает, является ли объект или переданный objectID контейнером
CODE
(isContainer:bool) reference.IsContainer objectID:ref

Edible (Съедобные)
Типы с этим свойством считаются едой.
Значения:
Is Food флаг - bool - является ли объект едой
Функции:
IsFood – возвращает, является ли объект едой
CODE
(isFood:bool) reference.IsFood objectID:ref
SetIsFood – задает, будет ли объект считаться едой
CODE
(nothing) reference.IsFood isFood:bool objectID:ref

Enchantable (Зачаровываемые)
Типы с этим свойством могут быть зачарованы.
Несколько замечаний: Зачарования и сила их эффектов могут быть присвоены объектам с возможностью зачарования. Тип Зачарования должен соответствовать объекту, которому оно присваивается. Также, при добавлении зачарования к объекту, убедитесь, что у объекта есть заряд, иначе зачарование работать не будет. Если объект ранее не был зачарован, вызовите SetObjectCharge, чтобы задать заряд объекта.
Значения:
Enchantment - ref - определенное зачарование на зачаровываемом предмете
Charge - long - максимальный заряд предмета
Current Charge - float - текущий заряд предмета
Функции:
GetEnchantment – возвращает зачарование объекта
CODE
(enchantment:ref) reference.GetEnchantment objectID:ref
SetEnchantment – задает зачарование объекту и возвращает предыдущее зачарование
CODE
(oldEnchantment:ref) reference.SetEnchantment newEnchantment:ref objectID:ref
RemoveEnchantment – удаляет зачарование с объекта и возвращает предыдущее зачарование
CODE
(oldEnchantment:ref) reference.RemoveEnchantment objectID:ref
GetObjectCharge – возвращает максимальный заряд объекта
CODE
(charge:long) reference.GetObjectCharge objectID:ref
SetObjectCharge – задает максимальный заряд объекта
CODE
(nothing) refernce.SetObjectCharge nuCharge:long objectID:ref
ModObjectCharge – изменяет максимальный заряд объекта вверх или вниз
CODE
(nothing) reference.ModObjectCharge modifyBy:float objectID:ref
GetEquippedCurrentCharge – возвращает текущий заряд объекта в указанном слоте
CODE
(charge:long) reference.GetEquippedCurrentCharge slot:short
SetEquippedCurrentCharge – задает текущий заряд объекта в указанном слоте
CODE
(nothing) reference.SetEquippedCurrentCharge nuCharge:long slot:short
ModEquippedCurrentCharge – изменяет текущий заряд объекта в указанном слоте вверх или вниз
CODE
(nothing) reference.ModEquippedCurrentCharge modifyBy:float slot:short

Equippable (Надеваемые)
Типы с этим свойством могут быть надеты персонажем.
Значения:
Slot - short - слот экипировки объекта
Функции:
GetEquipmentSlot – возвращает слот экипировки объекта
CODE
(slot:short) reference.GetEquipmentSlot objectID:ref
GetEquippedObject

Inventory (Инвентарь)

Типы с этим свойством могут храниться как в инвентаре, так и существовать как объект вне инвентаря.
Значения:
Object - long - базовый object id объекта (т.н. базовый объект)
Weapon Type- short - тип оружия
Weight - float - вес одного объекта этого типа
Gold Value - long - цена одного объекта этого типа. Это значение может несовпадать с отображаемым в интерфейсе, т.к. то может включать дополнительную стоимость зачарования.
Is Quest Item flag - bool - является ли тип квестовый предметом. Квестовые предметы нельзя активировать или выбросить.
Функции:
GetBaseObject – возвращает базовый объект
CODE
(objectID:ref) reference.GetBaseObject
GetType – возвращает тип кода для каждого типа объекта
CODE
(type:long) reference.GetType objectID:ref
IsWeapon – является ли объект оружием
CODE
(isWeapon:bool) reference.IsWeapon objectID:ref
IsAmmo – является ли объект амуницией
CODE
(isAmmo:bool) reference.IsAmmo objectID:ref
IsArmor – является ли объект броней
CODE
(isArmor:bool) reference.IsArmor objectID:ref
IsBook – является ли объект книгой
CODE
(isBook:bool) reference.IsBook objectID:ref
IsClothing – является ли объект одеждой
CODE
(isClothing:bool) reference.IsClothing objectID:ref
IsIngredient - является ли объект ингредиентом
CODE
(isIngredient:bool) reference.IsIngredient objectID:ref
IsKey - является ли объект ключом
CODE
(isKey:bool) reference.IsKey objectID:ref
IsAlchemyItem - является ли объект алхим.предметом
CODE
(isAlchemyItem:bool) reference.IsAlchemyItem objectID:ref
IsApparatus - является ли объект алхим.аппаратом
CODE
(isApparatus:bool) reference.IsApparatus objectID:ref
IsSoulGem - является ли объект камнем души
CODE
(isSoulGem:bool) reference.IsSoulGem objectID:ref
IsSigilStone - является ли объект камнем сигил
CODE
(isSigilStone:bool) reference.IsSigilStone objectID:ref
IsLight - является ли объект источником света
CODE
(isLight:bool) reference.IsLight objectID:ref
GetWeight – возвращает вес предмета
CODE
(weight:float) reference.GetWeight objectID:ref
SetWeight – задает вес предмета
CODE
(nothing) reference.SetWeight nuWeight:float objectID:ref
ModWeight – изменяет вес предмета вверх или вниз
CODE
(nothing) reference.ModWeight modifyBy:float objectID:ref
GetGoldValue – возвращает базовую цену объекта
CODE
(goldValue:long) reference.GetGoldValue objectID:ref
SetGoldValue – задает базовую цену объекта
CODE
(nothing) reference.SetGoldValue nuGoldValue:long objectID:ref
ModGoldValue – изменяет базовую цену объекта вверх или вниз
CODE

(nothing) reference.ModGoldValue modifyBy:float objectID:ref
IsQuestItem – возвращает, является ли предмет квестовым
CODE
(isQuestItem:bool) reference.IsQuestItem objectID:ref
SetQuestItem – задает, является ли предмет квестовым
CODE
(nothing) reference.SetQuestItem isQuestItem:bool objectID:ref

Magic (Магия)
Типы с этим свойством имеют группу магических эффектов
Значения:
Magic Item Type - short - тип магического предмета (Spell (Заклинание), Enchantment (Зачарование), AlchemyItem (Ахим.предмет), Ingredient (Ингредиент))
Effect Count - long - число магических эффектов в предмете
EffectItem Code - long - код магического эффекта
EffectItem Magnitude - long - сила магического эффекта
EffectItem Area - long - площадь, покрываемая магическим эффектом
EffectItem Duration - long - продолжительность магического эффекта в секундах
EffectItem Range - short - дистанция эффекта (Self (на себя), Touch (касание), Target (на цель))
EffectItem Actor Value - long - код для атрибута или навыки, затрагиваемого эффектом
Функции:
GetMagicItemType – возвращает тип магического предмета
CODE
(magicItemType:short) GetMagicItemType objectID:ref
(magicItemType:short) GetMIType objectID:ref

GetMagicItemEffectCount – возвращает число магических эффектов предмета
CODE
(count:long) GetMagicItemEffectCount objectID:ref
(count:long) GetMIECount objectID:ref
MagicItemHasEffect – возвращает, есть ли у магического предмета определенный эффект. Использует 4 буквенный код для маг.эффекта из КС.
CODE
(hasEffect:bool) MagicItemHasEffect effect:chars
(hasEffect:bool) MagicHasEffect effect:chars
MagicItemHasEffectCode - возвращает, есть ли у магического предмета определенный эффект. Использует значение типа long, возвращаемое функцией GetNthEffectItemCode или GetMagicEffectCode.
CODE
(hasEffect:bool) MagicItemHasEffectCode code:long
(hasEffect:bool) MagicHasEffectC code:long
GetNthEffectItemCode – возвращает код маг.эффекта указанного предмета
CODE
(code:long) GetNthEffectItemCode objectID:ref whichEffect:short
(code:long) GetNthEICode objectID:ref whichEffect:short
GetNthEffectItemMagnitude – возвращает силу определенного эффекта
CODE
(magnitude:long) GetNthEffectItemMagnitude objectID:ref whichEffect:short
(magnitude:long) GetNthEIMagnitude objectID:ref whichEffect:short
SetNthEffectItemMagnitude – задает силу определенного эффекта
CODE
(nothing) SetNthEffectItemMagnitude nuMagnitude:long objectID:ref whichEffect:short
(nothing) SetNthEIMagnitude nuMagnitude:long objectID:ref whichEffect:short
ModNthEffectItemMagnitude – изменяет силу определенного эффекта вверх или вниз
CODE
(nothing) ModNthEffectItemMagnitude modifyBy:float objectID:ref whichEffect:short
(nothing) ModNthEIMagnitude modifyBy:float objectID:ref whichEffect:short
GetNthEffectItemArea – возвращает зону покрытия эффекта
CODE
(area:long) GetNthEffectItemArea objectID:ref whichEffect:short
(area:long) GetNthEIArea objectID:ref whichEffect:short
SetNthEffectItemArea – задает зону покрытия эффекта
CODE
(nothing) SetNthEffectItemArea nuArea:long objectID:ref whichEffect:short
(nothing) SetNthEIArea nuArea:long objectID:ref whichEffect:short
ModNthEffectItemArea – изменяет зону покрытия эффекта вверх или вниз
CODE
(nothing) ModNthEffectItemArea modifyBy:float objectID:ref whichEffect:short
(nothing) ModNthEIArea modifyBy:float objectID:ref whichEffect:short

GetNthEffectItemDuration – возвращает продолжительность эффекта
CODE
(duration:long) GetNthEffectItemDuration objectID:ref whichEffect:short
(duration:long) GetNthEIDuration objectID:ref whichEffect:short
SetNthEffectItemDuration – задает продолжительность эффекта
CODE
(nothing) SetNthEffectItemDuration nuDuration:long objectID:ref whichEffect:short
(nothing) SetNthEIDuration nuDuration:long objectID:ref whichEffect:short
ModNthEffectItemDuration – изменяет продолжительность эффекта вверх или вниз
CODE
(nothing) ModNthEffectItemDuration modifyBy:float objectID:ref whichEffect:short
(nothing) ModNthEIDuration modifyBy:float objectID:ref whichEffect:short
GetNthEffectItemRange – возвращает дистанцию эффекта
CODE
(range:short) GetNthEffectItemrange objectID:ref whichEffect:short
(range:short) GetNthEIRange objectID:ref whichEffect:short
SetNthEffectItemRange – задает дистанцию эффекта
CODE
(nothing) SetNthEffectItemRange nuRange:short objectID:ref whichEffect:short
(nothing) SetNthEIRange nuRange:short objectID:ref whichEffect:short
GetNthEffectItemActorValue – возвращает значение актера определенного эффекта
CODE
(actorValueCode:long) GetNthEffectItemActorValue objectID:ref whichEffect:short
(actorValueCode:long) GetNthEIAV objectID:ref whichEffect:short
SetNthEffectItemActorValue – задает значение актера определенного эффекта
CODE
(nothing) SetNthEffectItemActorValue nuActorValue:long objectID:ref whichEffect:short
(nothing) SetNthEIAV nuActorValue:long objectID:ref whichEffect:short
RemoveNthEffectItem – удаляет определенный эффект
CODE
(nothing) RemoveNthEffectItem objectID:ref whichEffect:short
(nothing) RemoveNthEffect objectID:ref whichEffect:short

Named (Именованные)
Типы с этим свойством имеют задаваемые имена
Значения:
Name - string - отображаемое имя типа
Функции:
SetName – задает отображаемое имя объекта. SetName – специальная функция. Она определена, чтобы работать с Инвентарными предметами, но может быть использована на любой форме. Чтобы использовать ее на неинвентарном объекте, сначала нужно назначить objectID для ref. Для большинства форм название – часть базовой формы и изменение названия изменит его для всех копий предмета.
CODE
(nothing) reference.SetName name:string objectID:ref
Simple (Простые)
Типы с этим свойством имеют один путь к модели и одну иконку
Значения:
Model Path - string - путь к модели в формате NIF для предмета
Icon Path - string - путь к иконке в формате DDS для предмета
Функции:
SetModelPath – задает путь к модели для объекта
CODE
(nothing) reference.SetModelPath modelPath:string objectID:ref
SetIconPath - задает путь к иконке для предмета
CODE
(nothing) reference.SetIconPath iconPath:string objectID:ref

Wearable (Надеваемые)
Типы с этим свойством могут быть надетыми актерами и могут иметь различные модели и текстуры для мужчин и женщин.
Значения:
Slot - short - слот экипировки или слоты, занимаемые объектом
Male Model Path - string - путь к файлу в формате NIF для мужской или единой модели, когда та надета
Female Model Path - string - путь к файлу в формате NIF для женской модели, когда та надета
Male Ground Path - string - путь к файлу в формате NIF для мужской или единой модели, когда та брошена на землю
Female Ground Path - string - путь к файлу в формате NIF для женской модели, когда та брошена на землю
Male Icon Path - string - путь к иконке в формате DDS для мужской или единой модели
Female Icon Path - string - путь к иконке в формате DDS для женской модели
Функции:

GetEquipmentSlot – возвращает слот экипировки или слоты, занимаемые объектом
CODE
(slot:short) reference.GetEquipentSlot objectID:ref
SetEquipmentSlot – задает слот экипировки или слоты, занимаемые объектом
CODE
(nothing) reference.SetEquipmentSlot slot:short objectID:ref
SetMaleBipedPath – задает мужскую модель NIF, когда та надета
CODE
(nothing) reference.SetMaleModelPath modelPath:string objectID:ref
SetFemaleBipedPath - задает женскую модель NIF, когда та надета
CODE
(nothing) reference.SetFemaleModelPath modelPath:string objectID:ref
SetMaleGroundPath - задает мужскую модель NIF, когда та брошена
CODE
(nothing) reference.SetMaleGroundPath modelPath:string objectID:ref
SetFemaleGroundPath - задает женскую модель NIF, когда та брошена
CODE
(nothing) reference.SetFemaleGroundPath modelPath:string objectID:ref
SetMaleIconPath – задает иконку мужской одежде
CODE
(nothing) reference.SetMaleIconPath iconPath:string objectID:ref
SetFemaleIconPath - задает иконку женской одежде
CODE
(nothing) reference.SetFemaleIconPath iconPath:string objectID:ref

Oblivion Types (Типы Обливиона)

Типы Обливиона – это различные формы, доступные для заполнения в КС. Они сообщаются с реальными объектами или концептами в Обливионе. Типы – это собрания значений и функций, которые могут применяться к ним. Часто у них есть Свойства, которые инкапсулируют стандартные значения и функции между различными типами. Если у типа есть список свойств, то у этого типа есть все значения и функции этого свойства в дополнение к тому, что указано для данного типа напрямую.
Alchemy Item
Ammo
Armor
Clothing
Enchantment
Ingredient
Magic Effect Setting
NPC
Soul Gem
Spell
Weapon

Alchemy Item (Алхимический предмет)
Алхимические предметы – это зелья и яды.
Свойства: Edible, Inventory, Magic, Named, Simple
Значения:
Флаг Is Poison - bool - флаг определяет, считается ли алхимический предмет ядом. Яд – это алхимический предмет, в котором все эффекты отрицательные. Если имеется хоть один неотрицательный эффект, то алхимический предмет ядом не считается.
Функции:
IsPoison – возвращает, является ли алхим.предмет ядом.

CODE
(isPoison:bool) reference.IsPoison objectID:ref
Ammo (Амуниция)
Вся амуниция, типа стрел.
Свойства: Attacking, Enchantable, Equippable, Inventory, Named, Simple

Armor (Броня)
Броня – любой тип надеваемого объекта, который предоставляет защиту от повреждений
Свойства: Breakable, Enchantable, Inventory, Named, Wearable
Значения:
Armor Rating - long - значение защиты, даваемое броней
Armor Type - short - тип брони. Коды Типов Брони.
Функции:
GetArmorAR – возвращает рейтинг брони объекта
CODE

(armorRating:long) reference.GetArmorAR objectID:ref
SetArmorAR – задает рейтинг брони объекта
CODE
(nothing) reference.SetArmorAR nuArmorRating:long objectID:ref
ModArmorAR – изменяет рейтинг брони объекта вверх или вниз
CODE
(nothing) reference.ModArmorAR modifyBy:float objectID:ref
GetArmorType – возвращает 0, если броня легкая и 1, если тяжелая
CODE
(armorType:short) reference.GetArmorType objectID:ref
SetArmorType – задает тип брони: легкая или тяжелая
CODE
(nothing) reference.SetArmorType nuArmorType:short objectID:ref

Clothing (Одежда)
Одежда – любой надеваемый объект, который не дает защиты. Включает амулеты и кольца.
Свойства: Enchantable, Inventory, Named, Wearable

Enchantment (Зачарование)
Зачарование – группа магических эффектов, которые могут быть применены к зачаровываемым объектам.
Свойства: Named, Magic
Значения:
Enchantment Type - short тип объекта, к которому может быть применено зачарование
Charge - long заряд зачарования. По неизвестной и неясной причине – число иногда не соответствует значению, указанному в КС. Похоже, что в КС указывается цена и заряд вместе от значения Cost зачарованного предмета.
Cost - long размер магического заряда, затрачиваемого при каждом использовании зачарования
Функции:
GetEnchantmentType – возвращает тип зачарования
CODE
(enchantType:short) GetEnchantmentType objectID:ref
SetEnchantmentType – задает тип зачарования
CODE
(nothing) SetEnchantmentType enchantType:short objectID:ref
GetEnchantmentCharge – возвращает заряд зачарования.
CODE
(charge:long) GetEnchantmentCharge objectID:ref
SetEnchantmentCharge – задает заряд зачарования.
CODE
(nothing) SetEnchantmentCharge nuCharge:long objectID:ref
ModEnchantmentCharge – изменяет заряд зачарования
CODE
(nothing) ModEnchantmentCharge modifyBy:float objectID:ref
GetEnchantmentCost – возвращает цену магии за использование зачарования
CODE
(cost:long) GetEnchantmentCost objectID:ref
SetEnchantmentCost – задает цену магии за использование зачарования
CODE
(nothing) SetEnchantmentCost nuCost:long objectID:ref
ModEnchantmentCost – изменяет цену магии за использование зачарования вверх или вниз
CODE
(nothing) ModEnchantmentCost modifyBy:float objectID:ref
Ingredient (Ингредиент)
Ингредиенты – предметы, которые можно использовать для создания алхимических предметов
Свойства: Edible, Inventory, Magic, Named, Simple

Magic Effect Setting (Настройки Магического Эффекта)
Значения:
Code - chars или long код магического эффекта. В КС это 4 символьный код (как FIDG или Z001). Возвращаемое значение от функций OBSE – тип long с тем же численным значением.
Base Cost - float базовые затраты магии на эффект
School - short навык заклинания, контролирующий эффект
Projectile Speed - float скорость полета магического эффекта
Enchant Factor - float постоянный эффект фактора зачарования, помогающий определить максимальную силу эффекта при зачаровании
Barter Factor - float постоянный эффект фактора обмена, помогающий увеличить цену при зачаровании
Is Hostile флаг - bool определяет, является ли эффект негативным. Только негативные эффекты могут быть ядами.
Функции:
GetMagicEffectCode – возвращает код магического эффекта.

CODE
(magicEffectCode:long) GetMagicEffectCode effect:chars
(magicEffectCode:long) GetMECode effect:chars
GetMagicEffectBaseCost – возвращает базовые затраты магии на эффект
CODE
(baseCost:float) GetMagicEffectBaseCost effect:chars
(baseCost:float) GetMEBaseCost effect:chars
(baseCost:float) GetMagicEffectBaseCostC effect:long
(baseCost:float) GetMEBaseCostC effect:long
GetMagicEffectSchool – возвращает школу магического эффекта
CODE
(magicSchool:short) GetMagicEffectSchool effect:chars
(magicSchool:short) GetMESchool effect:chars
(magicSchool:short) GetMagicEffectSchoolC effect:long
(magicSchool:short) GetMESchoolC effect:long
GetMagicEffectProjectileSpeed – возвращает скорость полета магического эффекта
CODE
(projectileSpeed:float) GetMagicEffectProjectileSpeed effect:chars
(projectileSpeed:float) GetMEProjSpeed effect:chars
(projectileSpeed:float) GetMagicEffectProjectileSpeedC effect:long
(projectileSpeed:float) GetMEProjSpeedC effect:long
GetMagicEffectEnchantFactor – возвращает постоянный эффект фактора заклинания магического эффекта
CODE
(enchantFactor:float) GetMagicEffectEnchantFactor effect:chars
(enchantFactor:float) GetMEEnchant effect:chars
(enchantFactor:float) GetMagicEffectEnchantFactorC effect:long
(enchantFactor:float) GetMEEnchantC effect:long
GetMagicEffectBarterFactor – возвращает постоянный эффект фактора бартера магического эффекта
CODE
(enchantFactor:float) GetMagicEffectBarterFactor effect:chars
(enchantFactor:float) GetMEEBarter effect:chars
(enchantFactor:float) GetMagicEffectBarterFactorC effect:long
(enchantFactor:float) GetMEBarterC effect:long
IsMagicEffectHostile – возвращает, является ли магический эффект негативным
CODE
(isHostile:bool) IsMagicEffectHostile effect:chars
(isHostile:bool) IsMEHostile effect:chars
(isHostile:bool) IsMagicEffectHostileC effect:long
(isHostile:bool) IsMEHostileC effect:long

NPC
Свойства: Class, Container
Значения:
Equipped Items - NPC могут надевать и использовать предметы
Функции:
GetEquippedObject – возвращает базовый объект предмета из указанного слота экипировки
CODE
(objectID:ref) reference.GetEquippedObject slot:short
GetEquippedCurrentCharge
ModEquippedCurrentCharge
SetEquippedCurrentCharge
GetEquippedCurrentHealth
ModEquippedCurrentHealth
SetEquippedCurrentHealth
GetEquippedWeaponPoison
SetEquippedWeaponPoison
RemoveEquippedWeaponPoison
IsRefEssential – возвращает, является ли указанный актер essential
CODE
(isEssential:bool) reference.IsRefEssential
SetRefEssential – задает значение essential указанному актеру
CODE
reference.SetRefEssential isEssential:bool
HasSpell – возвращает, есть ли у указанного NPC указанное заклинание
CODE
(hasSpell:bool) reference.HasSpell spell:ref

ModActorValue2 – изменяет текущее значение без изменения базового значения. Действует как заклинание: отрицательные числа уменьшают значение, которое может быть восстановлено, а положительные числа восстанавливают значение без превышения максимального значения.
CODE
(nothing) reference.ModActorValue2 actorValueName:string value:long
GetActorLightAmount – возвращает значение в формате float количества света, падающего на актера, или 100, если актер не находится в высокой/средне-высокой обработке.
CODE
(lightAmount:float) reference.GetActorLightAmout
Soul Gem (Камень душ)
Свойства: Inventory, Named, Simple
Значения:
Soul Level – уровень души, находящейся в камне в настоящий момент
Capacity – максимальный уровень души, который может поместиться в камень
Функции:
GetSoulLevel – возвращает уровень души, заключенной в камне
CODE
(soulLevel:short) GetSoulLevel objectID:ref
GetSoulGemCapacity - возвращает максимальный уровень души, который может поместиться в камень
CODE
(soulLevel:short) GetSoulGemCapacity objectID:ref
Spell (Заклинание)
Свойства: Magic, Named
Значения:
Spell Type - short - тип заклинания. ТипЗаклинания.
Magicka Cost - long - затраты магии для кастования заклинания
Mastery Level - short - уровень мастерства, необходимый для кастования заклинания. Spell Mastery Levels.
Функции:
GetSpellType – возвращает тип заклинания
CODE
(spellType:short) GetSpellType objectID:ref
SetSpellType – задает тип заклинания
CODE
(nothing) SetSpellType nuType:short objectID:ref
GetSpellMagickaCost – возвращает затраты магии при кастовании
CODE
(magickaCost:long) GetSpellMagickaCost objectID:ref
SetSpellMagickaCost* - задает затраты магии при кастовании
CODE
(nothing) SetSpellMagickaCost nuMagickaCost:long objectID:ref
ModSpellMagickaCost* - изменяет затраты магии при кастовании вверх или вниз
CODE
(nothing) ModSpellMagickaCost modifyBy:float objectID:ref
GetSpellMasteryLevel – возвращает требуемый уровень мастерства
CODE
(masteryLevel:short) GetSpellMasteryLevel objectID:ref
SetSpellMasteryLevel* - задает требуемый уровень мастерства
CODE
(nothing) SetSpellMasteryLevel masteryLevel:short objectID:ref
* эти значения могут быть изменены, но не будут иметь эффекта, если заклинание помечено как AutoCalc. В настоящий момент нет способа определить, установлен ли флаг AutoCalc, или изменить его через скрипт.

Weapon (Оружие)
Свойства: Attacking, Breakable, Enchantable, Equippable, Inventory, Named, Simple
Значения:
Reach - float - расстояние от обладателя докуда достает оружие
Weapon Type - short - тип оружия
Poison - ref - яд, примененный к оружию
Функции:
GetWeaponReach – возвращает дистанцию поражения оружием
CODE
(reach:float) reference.GetWeaponReach objectID:ref
SetWeaponReach – задает дистанцию поражения оружием
CODE
(nothing) reference.SetWeaponReach nuReach:float objectID:ref
ModWeaponReach – изменяет дистанцию поражения оружием вверх или вниз
CODE
(nothing) reference.ModWeaponReach modifyBy:float objectID:ref

GetWeaponType – возвращает тип оружия
CODE
(weaponType:short) reference.GetWeaponType objectID:ref
SetWeaponType – задает тип оружия
CODE
(nothing) reference.SetWeaponType weaponType:short objectID:ref
GetEquippedWeaponPoison – возвращает яд, примененный к экипированному оружию. Вызывается на держателе оружия.
CODE
(poison:ref) reference.GetEquippedWeaponPoison
SetEquippedWeaponPoison – задает яд, примененный к экипированному оружию и возвращает предыдущий яд. Вызывается на держателе оружия.
CODE
(oldPoison:ref) reference.SetEquippedWeaponPoison nuPoison:ref
RemoveEquippedWeaponPoison – удаляет и возвращает яд, примененный к экипированному оружию. Вызывается на держателе оружия.
CODE
(oldPoison:ref) reference.RemoveEquippedWeaponPoison

General Functions (Основные функции)
IsDoor – возвращает, является ли вызывающий объект или передаваемый objectID дверью
CODE
(isDoor:bool) reference.IsDoor objectID:ref
IsFurniture - возвращает, является ли вызывающий объект или передаваемый objectID мебелью
CODE
(isFurniture:bool) reference.IsFurniture objectID:ref
IsActivator - возвращает, является ли вызывающий объект или передаваемый objectID активатором
CODE
(isActivator:bool) reference.IsActivator objectID:ref
GetPlayerSpell – возвращает objectID текущего заклинания игрока
CODE
(spell:ref) GetPlayerSpell
GetGameLoaded – возвращает 1,если игра была загружена с последнего вызова данной функции
CODE
(gameLoaded:bool) GetGameLoaded
GetOBSEVersion – возвращает номер версии OBSE
CODE
(obseVersion:long) GetOBSEVersion
GetParentCell – возвращает objectID родительской ячейки вызывающего объекта
CODE
(parentCell:ref) reference.GetParentCell
PrintToConsole – печатает сообщение в консоли в том же формате, что и Message
CODE
(nothing) PrintToConsole formatString:string var1 var2 etc
SetNumericGameSetting – задает определенное игровое значение, заданное переменной или прямым значением
CODE
(nothing) SetNumericGameSetting gameSettingName:string value:float
GetNumericINISetting – возвращает определенное значение из ini-файла
CODE
(setting:float) GetNumericINISetting iniSettingName:string
SetNumericINISetting - задает определенное значение в ini-файле, заданное переменной или прямым значением
CODE
(nothing) SetNumericINISetting iniSettingName:string value:float
Cloning Functions (Функции клонирования)
Функции клонирования особенные. Они объявлены как принимающие Инвентарные Объекты, так что любой инвентарный объект можно передать в них, используя его ID из КС. Однако, вы можете клонировать любую форму, присвоив ее ID переменной в формате ref, а затем передав эту переменную в CloneForm.
Функции:
CloneForm – создает и возвращает новый базовый объект, являющийся точной копией переданного ObjectID
CODE
(clonedForm:ref) CloneForm objectID:ref

Примеры:
CODE
ref clonedInventoryItem
ref clonedSpell
ref originalSpell
set clonedInventoryItem to CloneForm WeapSteelShortsword
set originalSpell to StandardCalmTouch1Novice

set clonedSpell to CloneForm originalSpell
IsClonedForm – возвращает, является ли переданный objectID клонированной формой, или нет. Клонированная форма сохраняется в файле сохранения. Примерами клонированных форм являются созданные игроками зелмй, заклинаний и зачарованных предметов.
CODE
(isCloned:bool) IsClonedForm objectID:ref

Flow Control Functions (Функции управления потоком)
Функции управления потоком позволяют внедрить циклы в скрипты. Т.к. в интерпретаторе Обливиона нет встроенной поддержки циклов, синтаксис этих функций может показаться странным. Первая функция, SaveIP или Label, запишет позицию инструкции после себя во внутренний список. Вторая функция, RestoreIP или Goto, совершает прыжок назад, к сохраненной позиции. Вот простой пример цикла:
CODE
```
begin onActivate
; объявляем переменную, чтобы хранить число итераций цикла
long var
set var to 0

; сохраняем текущий указатель на инструкцию
; эта запись говорит, что начало цикла прямо перед командой PrintToConsole
SaveIP; можно использовать Label

; печатает текущую итерацию
PrintToConsole "loop %f" var

; обновляем счетчик цикла
set var to var + 1

; мы хотим сделать лишь 3 итерации, так что проверяем счетчик
if var < 3
; прыгаем к сохраненной позиции
RestoreIP; можно испльзовать Goto
endif

; если мы здесь, то вышли из цикла
end
```

При активации скрипт выведет:
CODE
```
loop 0.0000
loop 1.0000
loop 2.0000
```

Чтобы поддержать вложенные циклы, команды SaveIP и RestoreIP принимают опциональный целочисленный параметр, определяющий, в какой "слот" сохранять инструкции. Если слот не указан, то по умолчанию он равен 0. Во внутреннем списке есть 256 слотов. Вот пример вложенных циклов:
CODE
```
begin onActivate
long var
long var2

set var2 to 0

SaveIP 0  ; Label 0
; начало внешнего цикла
set var to 0

SaveIP 1  ; Label 1
; начало внутреннего цикла
PrintToConsole "loop %f %f" var var2

set var to var + 1
if var < 3
; ветвление к началу внутреннего цикла
RestoreIP 1  ; Goto 1
endif
set var2 to var2 + 1
```

```
if var2 < 3
; ветвление к началу внешнего цикла
RestoreIP 0     ; Goto 0
endif
end
```

Внутренний цикл использует слот 1, внешний – слот 0.

Важные примечания:
Возможно создать бесконечный цикл, используя эти функции, которые заставят Обливион зависнуть. Заметьте также, что, хотя список мест технически сохраняется глобально, они имеют смысл только внутри того скрипта, где были сохранены, и должны расцениваться как неверные в конце каждого скрипта. Наконец, не используйте SaveIP/Label или RestoreIP/Goto в конструкциях 'set' или 'if'. (например, "if RestoreIP == 4"). Конечно, их безопасно использовать внутри тела конструкции, как показано в примерах, только не в самих конструкциях.

Реализация этих функций в OBSE v0009 была неверной, ведущей к нарушению работы с памятью и краху игры, когда цикл был больше 10 итераций. Релиз v0009a исправил это, но функции в этой версии должна считаться "бетой" и не использоваться в релизных модах. Эти функции начнут полноценно поддерживаться лишь начиная с версии OBSE v0010, так что проверяйте, что GetOBSEVersion возвращает 10 или выше перед использованием этих функций.
SaveIP – сохраняет расположение команды, следующей за командой SaveIP
CODE
(nothing) SaveIP slot:long
(nothing) Label slot:long
RestoreIP – пререйти к ранее сохраненному месту
CODE
(nothing) RestoreIP slot:long
(nothing) Goto slot:long

Console Functions (Консольные функции)

Некоторые из консольных команд были расширены в скриптовые функции. Во многих случаях их функциональность не до конца документирована, т.к. они не были предназначены для использования в скриптах, и при их использовании могут происходить странные вещи. Изменения не будут сохранены в файл сохранения. Перечень модифицированных консольных команд:
con_GetINISetting
con_HairTint
con_ModWaterShader
con_RefreshINI
con_RunMemoryPass
con_SetCameraFOV
con_SetClipDist
con_SetFog
con_SetGameSetting
con_SetGamma
con_SetHDRParam
con_SetImageSpaceGlow
con_SetINISetting
con_SetSkyParam
con_SetTargetRefraction
con_SetTargetRefractionFire
con_SexChange
con_ToggleDetection
con_WaterDeepColor
con_WaterReflectionColor
con_WaterShallowColor

Input Functions (Функции ввода)
IsKeyPressed – возвращает, нажата ли в данный момент определенная клавиша. Используются стандартные коды windows.
CODE
(isKeyPressed:bool) IsKeyPressed windowsKeyCode:long
IsKeyPressed2 - возвращает, нажата ли в данный момент определенная клавиша. Используются DX сканкоды.
CODE
(isKeyPressed:bool) IsKeyPressed2 dxScanCode:long
GetKeyPress – возвращает DX сканкод нажатой клавиши. Если нажата более, чем одна клавиша, используйте whichIndex, чтобы выбрать, какой код вернуть.
CODE

(keyPressed:long) GetKeyPress whichIndex:long
GetNumKeysPressed – возвращает число нажатых клавиш
CODE
(count:long) GetNumKeysPressed
DisableKey – отключает клавишу с определенным dx сканкодом
CODE
(nothing) DisableKey dxScanCode:long
EnableKey – включает клавишу с определенным dx сканкодом. Выключается DisableKey.
CODE
(nothing) EnableKey dxScanCode:long
HoldKey – держит клавишу с определенным dx сканкодом
CODE
(nothing) HoldKey dxScanCode:long
ReleaseKey – отпускает нажатую клавишу с определенным dx сканкодом
CODE
(nothing) ReleaseKey dxScanCode:long
TapKey – нажимает клавишу с определенным dx сканкодом один раз
CODE
(nothing) TapKey dxScanCode:long
HammerKey – поддельные нажатия каждый кадр клавиши с определенным dx сканкодом
CODE
(nothing) HammerKey dxScanCode:long
AHammerKey - поддельные нажатия каждый кадр в альтернативе к HammerKey клавиши с определенным dx сканкодом
CODE
(nothing) AHammerKey dxScanCode:long
UnHammerKey – останавливает нажатие клавиши с определенным dx сканкодом
CODE
(nothing) UnHammerKey dxScanCode:long
GetControl – возвращает dx сканкод клавиши, используемой для определенного управления
CODE
(dxScanCode:long) GetControl whichControl:short
GetAltControl - возвращает dx сканкод альтернативной клавиши, используемой для определенного управления
CODE
(dxScanCode:long) GetAltControl whichControl:short
GetMouseButtonPress – возвращает dx сканкод нажатой кнопки мыши. Если нажата более, чем одна кнопка, используйте whichIndex, чтобы выбрать, какой код вернуть
CODE
(dxScanCode:long) GetMouseButtonPress whichIndex:long
GetNumMouseButtonsPressed – возвращает число нажатых кнопок мыши.
CODE
(count:long) GetNumMouseButtonsPressed
DisableMouse – предотвращает перемещения курсора мыши.
CODE
(nothing) DisableMouse
EnableMouse – отключает DisableMouse
CODE
(nothing) EnableMouse
MoveMouseX – двигает мышь по горизонтали на указанное число пикселей.
CODE
(nothing) MoveMouseX pixels:long
MoveMouseY - двигает мышь по вертикали на указанное число пикселей
CODE
(nothing) MoveMouseY pixels:long
SetMouseSpeedX - двигает мышь по горизонтали на указанное число пикселей в секунду
CODE
(nothing) SetMouseSpeedX pixels:float
SetMouseSpeedY - двигает мышь по вертикали на указанное число пикселей в секунду.
CODE
(nothing) SetMouseSpeedY pixels:float

Math Functions (Математические функции)
Abs – возвращает абсолютное значение аргумента
CODE
(absoluteValue:float) abs arg:float
Ceil – возвращает ближайшее целое число больше аргумента
CODE
(ceil:float) ceil arg:float
Exp – возвращает e в степени аргумента

CODE
(exp:float) exp arg:float
Floor - возвращает ближайшее целое число меньше аргумента
CODE
(floor:float) floor arg:float
Log – возвращает натуральный логарифм числа
CODE
(log:float) log arg:float
Log10 – возвращает основу 10 логарифма числа
CODE
(log10:float) log10 arg:float
Pow – возвращает базу, увеличенную в несколько раз
CODE
(pow:float) pow base:float exponent:float
Rand – возвращает случайное число между min и max
CODE
(rand:float) rand min:float max:float
SquareRoot – возвращает корень квадратный из аргумента
CODE
(sqrt:float) squareroot arg:float
(sqrt:float) sqrt arg:float

Trigonometry Functions (Тригонометрические функции)
ACos – возвращает арккосинус аргумента. ACos и DACos используют градусы. RACose использует радианы.
CODE
(acos:float) acos arg:float
(acos:float) dacos arg:float
(acos:float) racos arg:float
ASin – возвращает арксинус аргумента. ASin и DASin используют градусы. RASin использует радианы.
CODE
(asin:float) asin arg:float
(asin:float) dasin arg:float
(asin:float) rasin arg:float
ATan – возвращает арктангенс аргумента. ATan и DATan используют градусы. RATan использует радианы.
CODE
(atan:float) atan arg:float
(atan:float) datan arg:float
(atan:float) ratan arg:float
ATan2 – возвращает арктангенсы от аргументов. ATan2 и DATan2 используют градусы. RATan2 использует радианы.
CODE
(atan2:float) atan2 arg1:float arg2:float
(atan2:float) datan2 arg1:float arg2:float
Cos – возвращает косинус угла. Cos и DCos используют градусы. RCos использует радианы.
CODE
(cos:float) cos arg:float
(cos:float) dcos arg:float
(cos:float) rcos arg:float
Cosh – возвращает гиперболический косинус угла. Cosh и DCosh используют градусы. RCosh использует радианы.
CODE
(cosh:float) cosh arg:float
(cosh:float) dco

7.6 Описание функций OBSE v0009a в алфавитном порядке.

A

AHammerKey

Синтаксис:
CODE
AHammerKey key

Сообщает игре в каждом втором фрейме, что игрок нажимает на определенную клавишу клавиатуры, и игра может соответствующим образом отреагировать.
Использует противоположный фрейм, как и ф-я HammerKey.
Прекращает использование UnHammerKey.

Эта функция использует скан-коды DirectX.
Примеры:
CODE
HammerKey 30;A
AHammerKey 32;D

The player will jerk left and right repeatedly (assuming default set-up).
Примечания:
Скан-коды DirectX обычно используются в шестнадцатеричном коде, но эта функция принимает десятичные значения.
Вы можете также задавать значения, большие 255, которые предназначены для управления мышью. При кодах от 256 до 263 вы можете работать с кнопками мыши. Коды 264 и 265 предназначены для колесика мыши.
Типовые DX-сканкоды клавиш:
CODE

Hex	Dec	Button
0x01	1	Escape
0x02	2	1
0x03	3	2
0x04	4	3
....		
0x09	9	8
0x0A	10	9
0x0B	11	0
0x10	16	Q
0x11	17	W
0x12	18	E
....		
0x17	23	I
0x17	24	O
0x19	25	P
0x1C	28	Enter
0x1D	25	Left Control
0x1E	30	A
0x1F	31	S
0x20	32	D
....		
0x24	36	J
0x25	37	K
0x26	38	L
0x29	41	~(Console)
0x2A	42	Left Shift
0x2C	44	Z
0x2D	45	X
0x2E	46	C
....		
0x30	47	B
0x31	48	N
0x32	49	M
0x36	54	Right Shift
0x37	55	NUM*
0x38	56	Left Alt
0x39	57	Spacebar
0x3A	58	Caps Lock
0x3B	59	F1
0x3C	60	F2
0x3D	61	F3
0x43	67	F8
0x44	68	F9

0x45	69	F10
0x46	70	Scroll-Lock
0x47	71	NUM7
0x48	72	NUM8
0x49	73	NUM9
0x4A	74	NUM-
0x4B	75	NUM4
0x4C	76	NUM5
0x4D	77	NUM6
0x4E	78	NUM+
0x4F	79	NUM1
0x50	80	NUM2
0x51	81	NUM3
0x52	82	NUM0
0x53	83	NUM.
0x57	87	F11
0x58	88	F12
0x9C	156	Num Enter
0x9D	157	Right Control
0xB5	181	NUM/
0xB8	184	Right Alt
0xC8	200	Up Arrow
0xCB	203	Left Arrow
0xCD	205	Right Arrow
0xD0	208	Down Arrow

Полный перечень скан-кодов можно найти здесь.
См. также: HammerKey, UnhammerKey, TapKey, HoldKey, ReleaseKey
Относится к типу: OBSE Input Functions

Abs

Синтаксис:
CODE
abs [float]

Примеры:
CODE
set n to abs -54

n = 54
CODE
set n to abs 54

n = 54
Функция abs возвращает абсолютное значение числа, т.е. результат всегда положителен.
Относится к типу: OBSE Math Functions

Acos

Синтаксис:
CODE
acos [float]

Примеры:
CODE
set n to (r / x)
set theta to acos n

Возвращает arccosine числа (т.е. угол, чьим косинусон он является)
Примечания:

Начиная с версии OBSE v0005 все тригонометрические функции принимают и возвращают значения в градусах. Версии функций, работающих с радианами, все еще доступны, если к ним добавить префикс с буквой "R", т.е. RAcos.
Эта функция может использоваться в математической формуле (a + acos B), однако математическая формула не будет компилироваться как аргумент для этой функции (acos (a + B) не работает).
acos a + b = (acos a) + b
См. также: sin, cos, tan, asin, atan, atan2
Относится к типу: OBSE Math Functions | OBSE Trigonometric Functions

Asin

Синтаксис:
CODE
asin [float]

Примеры:
CODE
set theta to asin (r / y)

Возвращает арксинус числа (т.е., возвращается угол, с которого был взят синус)
Примечания:
Начиная с версии OBSE v0005 все тригонометрические функции принимают и возвращают значения в градусах. Версии функций, работающих с радианами, все еще доступны, если к ним добавить префикс с буквой "R", т.е. RAsin.
Эта функция может использоваться в математической формуле (a + asin B), но математическая формула не компилируется, как аргумент для этой функции (asin (a + B) не работает).
asin a + b = (asin a) + b
sin a + b = (sin a) + b
См. также: sin, cos, tan, acos, atan, atan2
Относится к типу: OBSE Math Functions | OBSE Trigonometric Functions

Atan

Синтаксис:
CODE
atan [float]

Функция возвращает арктангенс числа (т.е., угол, от которого был вычислен тангенс).
Примеры:
CODE
set theta to atan n

(в отличие от asin и acos, где приведенный пример был значимым, этот пример - это просто пример)
Примечания:
Начиная с версии OBSE v0005 все тригонометрические функции принимают и возвращают значения в градусах. Версии функций, работающих с радианами, все еще доступны, если к ним добавить префикс с буквой "R", т.е. RAtan.
Эта функция может использоваться в математической формуле (a + atan B), но математическая формула не компилируется, как аргумент для этой функции (atan (a + B) не работает).
atan a + b = (atan a) + b
См. также: sin, cos, tan, asin, acos, atan2
Относится к типу: OBSE Math Functions | OBSE Trigonometric Functions

Atan2

Синтаксис:
CODE
atan2 [float1] [float2]

Пример:
CODE
set theta to atan2 y x

Функция atan2 возвращает угол между векторами (float2; float1).
Примечания:
Возвращает значение в диапазоне от 0 до 180 или от 0 до -180
Начиная с версии OBSE v0005 все тригонометрические функции принимают и возвращают значения в градусах. Версии функций, работающих с радианами, все еще доступны, если к ним добавить префикс с буквой "R", т.е. RAtan2.
Эта функция будет работать в математической формуле (c + atan2 a B), но если вы используете в качестве аргумента для этой функции математическую формулу, то возвращаемый результат будет неверным (например, результатом atan2 a (b + c) будет a+b+c).

См. также: sin, cos, tan, asin, acos, atan
Относится к типу: OBSE Math Functions | OBSE Trigonometric Functions

C

Ceil

Функция Ceil дает те же результаты, что и приведённые в этой статье wiki примеры: Ceiling and Floor
Возвращает ближайшее целое число, большее указанного в виде параметра функции Ceil.
Синтаксис:
CODE
ceil [float]

Примеры:
CODE
set n to ceil 5.234

n = 6
См. также: floor
Относится к типу: OBSE Math Functions

CloneForm

Функция CloneForm относится к особым функциям клонирования.
Они объявлены как принимающие Инвентарные Объекты, так что любой инвентарный объект можно передать в них, используя его ID из КС. Однако, вы можете клонировать любую форму, присвоив ее ID переменной в формате ref, а затем передав эту переменную в CloneForm.
Функция CloneForm создает и возвращает новый базовый объект, являющийся точной копией переданного ObjectID

Синтаксис:
CODE
[nuObjectID] CloneForm [objectID]

Эта функция создает точный дубликат указанного в виде параметра ObjectID базового объекта и возвращает его новый nuObjectID. Клонированные формы сохраняются в игре. Клонирование объекта позволяет вам делать любые изменения на созданном объекте, не затрагивая его базовый (исходный) объект.
CODE
ref clone
ref originalObject

set originalObject to myReference.GetBaseObject
set clone to CloneForm originalObject; клонирование базового объекта в ref-переменную Clone
player.placeAtMe clone 1 0 0; перенесение копии в расположение игрока

CODE
ref clonedInventoryItem
ref clonedSpell
ref originalSpell
set clonedInventoryItem to CloneForm WeapSteelShortsword
set originalSpell to StandardCalmTouch1Novice
set clonedSpell to CloneForm originalSpell

Эта функция работает с любыми игровыми типами (объекты, заклинания, зачарования, NPCs).
Если функция вызывается на объекте инвентаря (оружие, броня, источники света, яды, книги, ингредиенты и т.п.), вы, возможно, передаете его EditorID из конструктора непосредственно в функцию.
Для любого другого типа вы должны первым делом установить ref-переменную
для соответствующего ObjectID перед вызовом функции CloneForm.
Примечания: У клонированных персонажей (NPC) в инвентаре нет никаких предметов, он пуст.
Относится к типу: OBSE Item Functions

Cos

Более известна как функция "Cosine" (косинус).
Возвращает косинус угла.
Синтаксис:
CODE
cos [float]

Примеры:
CODE
set x to r * (cos theta)

Примечания:
Начиная с версии OBSE v0005 все тригонометрические функции принимают и возвращают значения в градусах. Версии функций, работающих с радианами, все еще доступны, если к ним добавить префикс с буквой "R", т.е. RCos).
Эта функция будет работать в математической формуле (a + cos B), но если вы используете в качестве аргумента для этой функции математическую формулу, то эта функция компилироваться не будет (например, (cos (a + B)).
cos a + b = (cos a) + b
См. также: sin, tan, asin, acos, atan, atan2
Относится к типу: OBSE Math Functions | OBSE Trigonometric Functions

Cosh

Синтаксис:
CODE
cosh [float]

Примеры:
CODE
set x to cosh theta

Функция Cosh возвращает гиперболический косинус угла.
Примечания:
Начиная с версии OBSE v0005 все тригонометрические функции принимают и возвращают значения в градусах. Версии функций, работающих с радианами, все еще доступны, если к ним добавить префикс с буквой "R", т.е. RCosh).
Эта функция будет работать в математической формуле (a + cosh B), но если вы используете в качестве аргумента для этой функции математическую формулу, то эта функция компилироваться не будет (например, (cosh (a + B)).
cosh a + b = (cosh a) + b
См. также: sin, cos, tan, asin, acos, atan, sinh, tanh
Относится к типу: OBSE Math Functions | OBSE Trigonometric Functions

D

DisableKey

Синтаксис:
CODE
DisableKey key

Пример:
CODE
DisableKey 20;T

Функция DisableKey запрещает использование клавиши клавиатуры с указанным в десятичном исчислении скан-кодом (key). Для возврата используйте функцию EnableKey.
Prevents resting (assuming default set-up).
Примечания:
Скан-коды DirectX обычно используются в шестнадцатеричном коде, но эта функция принимает числа в десятичном исчислении.
Вы можете также задавать значения, большие 255, которые предназначены для управления мышью. При кодах от 256 до 263 вы можете работать с кнопками мыши. Коды 264 и 265 предназначены для колесика мыши.
Функция IsKeyPressed будет работать с заблокированными клавишами, но функция IsKeyPressed2 - нет, разве что функция TapKey была вызвана в предыдущем фрейме и все еще работает. Функции HoldKey, HammerKey и AHammerKey также не будут работать, пока клавиша заблокирована, но начнут работать немедленно после того, как будет запущена функция EnableKey, если их конечно не заблокируют перед этим другие функции.
Наиболее часто используемые сканкоды приведены в описании функции AHammerKey.
Полный список скан-кодов можно посмотреть здесь.
См. также: EnableKey
Относится к типу: OBSE Input Functions

DisableMouse

Функция DisableMouse предотвращает перемещение мыши, а также ее кнопки. Чтобы разблокировать мышь, используйте функцию EnableMouse.
Синтаксис:
CODE

DisableMouse

См. также: EnableMouse, SetMouseSpeedX, SetMouseSpeedY, MoveMouseX, MoveMouseX
Относится к типу: OBSE Input Functions

E

EnableKey

Синтаксис:
CODE
EnableKey key

Функция EnableKey разрешает использовать заблокированную ранее функцией DisableKey клавишу (key). Код клавиши необходимо указывать в десятичном исчислении.
Пример:
CODE
EnableKey 20;T

Разрешаем использование клавиши "T".
Примечания:
Скан-коды DirectX обычно используются в шестнадцатеричном коде, но эта функция принимает числа в десятичном исчислении.
Вы можете также задавать значения, большие 255, которые предназначены для управления мышью. При кодах от 256 до 263 вы можете работать с кнопками мыши. Коды 264 и 265 предназначены для колесика мыши.
Наиболее часто используемые сканкоды приведены в описании функции AHammerKey.
Полный список скан-кодов можно посмотреть здесь.
См. также: DisableKey
Относится к типу: OBSE Input Functions

EnableMouse

Синтаксис:
CODE
EnableMouse

Функция EnableMouse рарешает работу мыши, заблокированную ранее функцией DisableMouse.
См. также: DisableMouse, SetMouseSpeedX, SetMouseSpeedY, MoveMouseX, MoveMouseX
Относится к типу: OBSE Input Functions

Exp

Синтаксис:
CODE
exp [float]

Пример:
CODE
set n to exp n

Функция exp (экспонента) возвращает основание e, возведенное в указанную степень (float). Функция является обратной по отношению к функции натурального логарифма log.
См. также: log
Относится к типу: OBSE Math Functions

F

Floor

Функция Floor возвращает ближайшее целое число, меньшее указанного в виде параметра (float).
Синтаксис:
CODE
floor [float]

Пример:
CODE
set n to floor 5.784

n = 5
Примечание: Функция Floor делает то же, что и Ceiling and Floor
См. также: Ceil
Относится к типу: OBSE Math Functions

G

GetActorLightAmount

Синтаксис:
CODE
[Ref.]GetActorLightAmount

Функция GetActorLightAmount должна вызываться с помощью ref-переменной (Ref), указывающей на актера. Тип возвращаемой переменной - вещественное. Функция возвращает количество света, освещающее в данный момент копию актера. Большее значение означает и большую освещенность. Освещенность колеблется в диапазоне от "0.00" (полная темнота) до "100.00" (очень ярко).
Относится к типу: OBSE Reference Functions | OBSE Actor Functions

GetAltControl

Garin: В wiki эта функция описана совершенно невнятно. Поэтому я несколько доработал описание.
Функция GetAltControl возвращает номер кнопки мыши (MouseButtonID), которой назначен указанный в виде параметра игровой код ControlId (не путайте с DX-сканкодами!).
Синтаксис:
CODE
GetAltControl ControlId

Примеры:
CODE
set mouseAttackButton to GetAltControl 4; кнопка атаки
set mouseBlockButton to GetAltControl 7; кнопка блока
set mouseRestButton to GetAltControl 16; кнопка отдыха

Примечания:
Эта функция возвращает число, которое соответствует формуле (255 + MouseButtonID*256). Левая кнопка мыши - ID "0", правая - ID "1", средняя - ID "2" и т.д. вплоть до максимальной с ID "7".
Если эта кнопка не привязана к управлению в игре или управление недействительно, эта функция возвратит значение 65535 (-1, если считывать значение в короткую целочисленную переменную (short)).
Если вы хотите знать, какая клавиша клавиатуры, связанная с управлением, была нажата, используйте функцию GetControl.
Чтобы преобразовать значение, возвращенное GetAltControl, в DX-сканкод, пригодный для использования с функцией IsKeyPressed2, включите в свой скрипт код, подобный этому:
CODE
short button
set button to getAltControl 4; attack button
if (button != -1); определяем, назначено ли действие
; "атака" какой-нибудь кнопке мыши
set button to (button - 255) / 256 + 256
endif

Информация:
MouseButtonID. Как известно, мышки бывают разные. У одних имеются всего две кнопочки, другие - трехкнопочные со скроллом, а третьи - не мышь, а целая игровая консоль... Вы можете назначить управление всем имеющимся у вас "мышиным" кнопкам, с тем, чтобы в игре можно было открывать определенные окна или совершать некоторые действия, просто нажав на соответствующую кнопку. Кнопки мыши имеют каждая свой ID-код (MouseButtonID). Левая кнопка мыши имеет значение ID "0", правая - ID "1", средняя - ID "2" и т.д. вплоть до максимальной со значением ID "7". Действия, которые можно назначить этим кнопкам (не только кнопкам мыши, но и клавишам клавиатуры), также имеют свой ID-код (Control IDs).
DX-сканкоды. В отличие от игровых ID-кодов, DX-сканкоды возвращают стандартный код нажатой клавиши DirectInput и от игры не зависят. Но зато с помощью функций OBSE их можно с успехом использовать для определения нажатой в игре клавиши и использовать в своих модах. Что такое сканкоды DirectInput, вы можете прочитать здесь:
http://coop.chuvashia.ru/kartuzov/mgr/Book...l-fil=wgp06.htm
Таблица соответствия Control IDs приведена в Приложении 4: Игровые ID управления (Control IDs)
См. также: GetControl
Относится к типу: OBSE Input Functions

GetArmorAR

Синтаксис:
CODE
[ref.]GetArmorAR [objectID]

Функция GetArmorAR возвращает рейтинг брони вызывающей копии (ref) либо базового объекта (ObjectID). Базовый объект ObjectID получает превосходство, если вызваны оба.
Значение, возвращаемое этой функцией, есть рейтинг брони, определенный в конструкторе, и умноженный на 100. Навык владельца во внимание не принимается.
Примечание: существует эквивалент - не используемая более функция GetObjectValue, которая работает с аргументом 150.
Относится к типу: OBSE Item Functions

GetBaseObject

Синтаксис:
CODE
[Ref.]GetBaseObject

Функция GetBaseObject должна вызываться на копии (Ref). Возвращает указатель (ref-переменная) на базовый объект.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

GetControl

Синтаксис:
CODE
GetControl ControlId

Функция GetControl возвращает DirectX-сканкод клавиши клавиатуры, которая привязана к указанному в виде параметра ControlId действию для управления игрой (см. список Control ID в описании функции GetAltControl).
Примеры:
CODE
set ForwardKey to GetControl 0
set CastKey to GetControl 7

Примечания:
Эта функция возвращает DirectX-сканкоды в диапазоне от 0 до 255. Если ни одна из клавиш к игровому управлению не привязана или управление отключено, функция возвратит значение 65535 (-1, если считывать как короткую целочисленную переменную).
Если вам требуется узнать номер кнопки мыши, которая привязана к управлению в игре (открыть диалоговое окно или выполнить какое-либо действие), используйте функцию GetAltControl.
Если переназначить управление, функция GetControl (начиная с v0007) не будет возвращать новое значение до тех пор, пока вы не выйдете и снова не загрузите игру.
ID управления (Control IDs) приведены в Приложении 4: Игровые ID управления (Control IDs)
Наиболее часто используемые DX-сканкоды приведены в Приложении 3: Сканкоды DX
См. также: GetAltControl
Относится к типу: OBSE Input Functions

GetCurrentValue

Синтаксис:
CODE
[ref.]GetCurrentValue ValueType
[ref.]GetCV ValueType

Функция GetCurrentValue возвращает текущее значение запрашиваемого типа величины ValueType из дополнительной информации о вызывающей копии ref.
Функция возвращает "0", если тип указанного значения не соответствует вызывающей копии (например, для оружия запрашивается рейтинг брони, или для одежды - ее изношенность). Возвращает базовое значение, если оно не изменялось из базы. В настоящий момент только здоровье, нагрузка и яд будут возвращать различные значения по сравнению с GetOV.
Примеры:
CODE
scn ScriptonSentientSword
Begin Gamemode
If GetCV 2 < 50
Message "Чувствующий меч телепатически обращается к вам с просьбой о ремонте."
Endif
end

Типы возвращаемых величин приведены в “Приложение 1: Возвращаемые типы величин”

См. также: GetObjectValue, GetEquippedCurrentValue, GetEquippedObjectValue
Относится к типу: OBSE Item Functions | OBSE Reference Functions

GetEquippedCurrentValue

Функция GetEquippedCurrentValue возвращает текущее значение запрашиваемого типа величины ValueType из дополнительной информации о вызывающей копии ref.
Функция возвращает "0", если тип указанного значения не соответствует вызывающей копии (например, для оружия запрашивается рейтинг брони, или для одежды - ее здоровье). Возвращает базовое значение, если оно не изменялось из базы.
В настоящий момент только здоровье, нагрузка и яд будут возвращать различные значения по сравнению с GetOV.
Синтаксис:
CODE
GetEquippedCurrentValue ValueType SlotID
GetECV Valuetype SlotID

Пример:
CODE
if player.getecv 2 16 <= 0.3 * player.geteov 2 16
message "Качество вашего текущего оружия снизилось до 30%. Отремонтируйте его. "
endif

Value Types:
CODE
Диапазон - тип величины
Range - Value Types

000-099 - Общие значения (Common Values)
100-149 - Оружие и стрелы (Weapon and Ammo)
150-199 - Броня (Armor)
200-209 - Камни душ (Soul Gem)
210-219 - Ингредиенты (Ingredient)
220-229 - Алхимические предметы (Alchemy Item)

Полный перечень возвращаемых величин приведен в “Приложение 1: Возвращаемые типы величин”
Перечень всех доступных для экипировки слотов приведен в “Приложение 2: Слоты ID для экипировки (Slot IDs)”
См. также: GetEquippedObjectValue, GetCurrentValue, GetObjectValue
Относится к типу: OBSE Item Functions | OBSE Reference Functions

GetEquippedObject

Синтаксис:
CODE
GetEquippedObject [slot]

Функция GetEquippedObject должна вызываться на указателе. Возвращает ObjectID объекта, размещенного в указанном для экипировки слоте (slot).
Примеры:
CODE
ref Weapon
set Weapon to player.GetEquippedObject 16

ref UpperBodyArmor
set UpperBodyArmor to player.GetEquippedObject 2

Сохраняет в ref-переменных ID-коды оружия и экипированного на игроке объекта в указанных слотах.
Перечень всех доступных для экипировки слотов приведен в “Приложение 2: Слоты ID для экипировки (Slot IDs)”
Примечания:
До релиза OBSE v0006 эта функция была называлась GetEquipmentSlotType.
Относится к типу: OBSE Inventory Functions | OBSE Reference Functions

GetEquippedObjectValue

Синтаксис:
CODE
GetEquippedObjectValue ValueType SlotID
GetEOV Valuetype SlotID

Функция GetEquippedObjectValue возвращает для вызывающего объекта значение указанного типа величины (ValueType), размещенного в слоте SlotID предмета. Функция возвратит «0», если тип значения не соответствует запрашиваемой величине (например, для оружия запрашивается степень защиты брони, или состояние изношенности для одежды).
Пример:
CODE
float weaponspeed
set weaponspeed to player.geteov 102 16

В приведенном примере в переменной "weaponspeed" сохраняется значение скорости игрока, экипированного оружием указанного типа.
Полный перечень возвращаемых величин приведен в Приложении 1 “Возвращаемые типы величин (ValueType)”
Перечень всех доступных для экипировки слотов приведен в Приложении 2 “Слоты ID для экипировки (Slot IDs)”
См. также: GetEquippedCurrentValue, GetObjectValue, GetCurrentValue
Относится к типу: OBSE Item Functions | OBSE Reference Functions

GetGameLoaded

Синтаксис:
CODE
if (GetGameLoaded)

Функция GetGameLoaded возвращает истину (“1”), если с момента последнего вызова этой функции сохраненная игра была загружена.
Примечание:
Каждый вызов этой функции в вызываемом скрипте сбрасывает флаг. Все скрипты, которые будут вызваны после этого, не дадут результата.
Относится к типу: OBSE Flow Control Functions | OBSE Miscellaneous Functions

GetInventoryObject

Синтаксис:
CODE
GetInventoryObject index

Функция GetInventoryObject должна вызываться на копии. Возвращает тип предмета (indexth), размещенный в контейнере (инвентаре) под указанным индексом (index).
Примеры:
CODE
ref itemType; объявляем ref-переменную itemType
set itemType to (player.GetInventoryObject 0)
container.additem itemType 1

В приведенном примере функция возвращает предмет, который в инвентаре игрока был размещен первым, после чего в другой контейнер будет добавлен один предмет того же типа.
Примечания:
Объекты сохраняются в том в порядке, в котором они размещались. Если Вы приобрели меч, затем лук и после этого щит, то вызов “GetInventoryObject 0” возвратит меч, если указать в качестве параметра функции “1”, то будет возвращен лук, если указан параметр “2”, то будет возвращен щит.
Предметы помнят порядок, в котором они размещались, поэтому, например, если вы бросите лук на землю, затем подберете его, после чего вызовете GetInventoryObject 1, то будет возвращен все тот же лук, а щит будет находиться под индексом 2.
Так же, как и функция GetNumItems, функция GetInventoryObject работает с уникальными предметами, если под индексом 4 числились стрелы, и их у вас насчитывалось 10 шт, то вызов GetInventoryItemType 3 возвратит тип “железные стрелы” (все правильно – нумерация начинается с 0, поэтому индекс 3 соответствует 4-му предмету). В этом случае вызов GetInventoryItemType 4 возвратит следующий по списку предмет, но отнюдь не следующую железную стрелу.
Вплоть до OBSE v0006 эта функция была известна как GetInventoryItemType.
Относится к типу: OBSE Inventory Functions | OBSE Reference Functions

GetKeyPress

Синтаксис:
CODE
GetKeyPress id

Функция GetKeyPress возвращает DX-сканкод клавиши клавиатуры, которая нажата в данный момент. Если нажато несколько, то определить, что среди нажатых клавиш находится нужная можно, если указать в виде параметра ее ID.
Примечания:
Эта функция не определяет коды нажатых кнопок мыши.
Эта функция возвращает сканкод клавиши в течение всего периода, пока клавиша нажата, а не только в первом фрейме.

Если никакие клавиши не нажаты или если указанный в виде параметра ID не совпадает с нажатыми клавишами, будет возвращено значение 65535.
Перечень сканкодов смотрите в Приложении 3 OBSE: Сканкоды DX
См. также: GetNumKeysPressed
Относится к типу: OBSE Input Functions

GetNumItems

Синтаксис:
CODE
GetNumItems

Функция GetNumItems должна вызываться на копии. Возвращает количество уникальных предметов в контейнере или инвентаре.
Примечания:
Отметьте, что возвращается количество только уникальных типов предметов в инвентаре, например, 10 стрел будут считаться одним уникальным объектом.
Золото также будет считаться одним предметом, независимо от его количества.
Экипированные объекты и такие же, находящиеся в инвентаре, также будут считаться как один уникальный предмет.
Относится к типу: OBSE Inventory Functions | OBSE Reference Functions

GetNumKeysPressed

Синтаксис:
CODE
GetNumKeysPressed

Функция GetNumKeysPressed возвращает общее количество нажатых в данный момент клавиш клавиатуры.
Примечания:
Эта функция не работает с кнопками мыши.
Большинство клавиатур могут отслеживать только ограниченное количество одновременно нажатых клавиш, поэтому могут возвращать некорректные комбинации нажатых клавиш. Эта функция всегда точно отразит, сколько клавиш нажато, чтобы можно было попытаться правильно отреагировать на большое количество нажатых клавиш. В Обливионе, например, можно видеть ситуацию с нажатыми клавишами, но не всегда определить, что на самом деле замыслил игрок.
См. также: GetKeyPress
Относится к типу: OBSE Input Functions

GetOBSEVersion

Синтаксис:
CODE
GetOBSEVersion

Функция GetOBSEVersion возвращает установленную на компьютере игрока версию OBSE. Может быть использована для проверки и обеспечения совместимости с установленными модами.
Пример:
CODE
if (GetOBSEVersion < 5)
MessageBox "Требуется Oblivion Script Extender v0005 или выше."
Return
endif

Примечания:
В конструкциях If…Endif для определения более старых версий всегда используйте проверку «менее, чем», иначе использование «не равно» может “запереть” вас на текущей версии OBSE.
Относится к типу: OBSE Debug Functions

GetObjectType

Функция GetObjectType возвращает тип объекта и может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref). Если указать оба, то предпочтение будет отдано указателю на объект.
Синтаксис:
CODE
GetObjectType [ObjectID]

Функция GetObjectType возвращает для указанного в виде параметра ObjectID объекта его тип.
CODE

[ref.]GetObjectType

Вызывается на указателе на объект (ref). Возвращается тип ID вызывающей копии.
Возвращаемые коды типов объектов (Type Codes Returned)

CODE

0 None
1 TES4
2 Group
3 GMST
4 Global
5 Class
6 Faction
7 Hair
8 Eyes
9 Race
10 Sound
11 Skill
12 Effect
13 Script
14 LandTexture
15 Enchantment
16 Spell
17 Birthsign
18 Activator
19 Apparatus
20 Armor
21 Book
22 Clothing
23 Container
24 Door
25 Ingredient
26 Light
27 Misc
28 Static
29 Grass
30 Tree
31 Flora
32 Furniture
33 Weapon
34 Ammo
35 NPC
36 Creature
37 Creature Leveled List
38 Soul Gem
39 Key
40 Alchemy Item
41 Sub Space
42 Sigil Stone
43 Item Leveled List
44 SNDG
45 Weather
46 Climate
47 Region
48 Cell
49 REFR
50 ACHR
51 ACRE
52 Path Grid
53 Worldspace
54 Land
55 TLOD
56 Road
57 Dialog
58 Dialog Info
59 Quest
60 Idle
61 Package
62 Combat Style

63 Load Screen
64 Spell Leveled List
65 ANIO
66 Water Form
67 Effect Shader
68 TOFT

Примечание: В данной версии OBSE отнюдь не все коды возвращаются. Однако в дальнейшем, по мере выхода следующих версий OBSE, все они станут доступными.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

GetObjectValue

Синтаксис:
CODE
GetObjectValue ValueType [ObjectID]
GetOV Valuetype [ObjectID]
ref.GetObjectValue ValueType
ref.GetOV ValueType

Функция GetObjectValue возвращает значение указанного в виде параметра ValueType типа величины. Функция может вызываться как в отношении указанного в виде необязательного параметра ID объекта (ObjectID), так и для указателя на объект (ref). Функция возвратит “0”, если тип запрашиваемой величины не соответствует вызывающему объекту (для оружия запрашивается степень защиты, или для одежды – ее изношенность)
Примеры:
CODE
ref equippedweapon
float weaponspeed
short weapontype
set equippedweapon to player.GetEquippedObject 16
set weaponspeed to getov 102 equippedweapon
set weapontype to getov 103 equippedweapon

Вызывается из скрипта, “повешенного” на объект
CODE
float weaponspeed
short weapontype
begin OnEquip
set weaponspeed to GetOV 102
set weapontype to GetOV 103
end

Типы возвращаемых величин приведены в Приложении 1: “Возвращаемые типы величин (OBSEValueTypes)”
См. также: GetCurrentValue, GetEquippedObjectValue, GetEquippedCurrentValue
Относится к типу: OBSE Item Functions | OBSE Reference Functions

GetParentCell

Синтаксис:
CODE
GetParentCell

Функция GetParentCell возвращает ячейку, содержащую вызывающую копию объекта.
Вызывается с помощью указателя на объект (ref).
Пример:
CODE
scn CellChangedScript
float fQuestDelayTime
ref CellLastFrame
Begin GameMode
set fQuestDelayTime to 0.0001
if (CellLastFrame != player.GetParentCell)
set CellChanged to 1
set CellLastFrame to player.GetParentCell
else
set CellChanged to 0
endif
End

Этот скрипт имитирует функцию CellChanged из игры Morrowind. В нашем случае CellChanged будет глобальной короткой целочисленной переменной.
Примечание:
В сопровождающем файле commands.txt к OBSE v0004 эта функция именовалась как GetPlayerCell.
Относится к типу: OBSE Miscellaneous Functions | OBSE Reference Functions

GetPlayerSpell

Синтаксис:
CODE
GetPlayerSpell

Функция GetPlayerSpell возвращает активное заклинание игрока.
Returns the player's active spell.
Пример:
CODE
ref spell
set spell to GetPlayerSpell
player.Cast spell

Примечания:
Удаление активного заклинания нужно производить осторожно. Разработчикам OBSE пока не известен полный список заклинаний игрока, поэтому, если вы удаляете активное заклинание, функция будет возвращать “0” до тех пор, пока игрок не выберет новое активное заклинание. В будущем эта проблема будет решена, как только недостающая информация будет найдена.
Расовые магические способности могут быть удалены таким же образом.
В версии OBSE v0004 эта функция именовалась как GetActiveSpell. Функция была переименована, чтобы название этой функции OBSE соответствовало названию функции Обливиона SelectPlayerSpell.
См. также: SelectPlayerSpell
Относится к типу: OBSE Magic Functions | OBSE Hero Functions

Con_GetINISetting

Синтаксис:
CODE
con_GetINISetting "Setting:Subsection"

Функция Con_GetINISetting возвращает желаемое значение параметра, прописанного в Oblivion.ini и указанного кавычках в вызываемой функции в виде параметра "Setting:Subsection". Название соответствует названию соответствующего параметра в ini-файле. Подраздел является текстом в скобках и появляется выше группы, к которой указанная установка принадлежит.
Пример:
CODE
con_GetINISetting "bCrossHair:Gameplay"

В данном примере функция возвращает значение 0 или 1, указывающее на то, включен или нет игровой курсор в базовом ini-файле.
Примечания:
Эта функция позволяет вручную изменить уставку только в тех случаях, когда ее имя начинается с “b”, ”i” или “f”. Это связано с тем, что игре нужно знать, к какому типу относятся переменные, с которыми она оперирует. Установки, начинающиеся с других букв, может быть переменными любого типа.
Из-за особенностей программного обеспечения Wiki название этой функции отображается некорректно – пропадает символ подчеркивания. Правильное название - Con_GetINISetting.
Эта функция идентична консольной команде GetINISetting и при вызове в скрипте ведет себя так, как будто Вы вызвали ее с консоли.
Эта функция не предназначалась для работы в скриптах, поэтому не будет работать так, как вам бы хотелось.
Относится к типу: OBSE Console Functions

GoTo

Синтаксис:
CODE
GoTo [labelID]

Функция GoTo позволяет в процессе выполнения скрипта перейти на метку labelID, определенную в пределах этого же скрипта. Работает аналогично функции RestoreIP. Для установки метки вы можете использовать функции OBSE Label или SaveIP. Явное указание необязательного параметра labelID может использоваться при использовании нескольких меток в пределах скрипта или организовать вложенные ветвящиеся циклы.

Пример:
CODE
begin onActivate
; объявляем переменную, в которой будем отслеживать количество выполненных циклов
long var
set var to 0
; сохраняем текущие установки указателя
; записываем начало цикла, который находится непосредственно перед командой PrintToConsole
SaveIP; устанавливаем метку, можно также было использовать Label.
; выводим текущий отсчет цикла
PrintToConsole "loop %f" var
; обновляем счетчик цикла
set var to var + 1
; нам нужно выполнить цикл всего три раза, поэтому проверяем состояние счетчика
if var < 3
; переходим на сохраненную метку
RestoreIP; здесь можно использовать функцию GoTo, если ранее использовалась Label
endif
; если мы дошли в это место, значит мы закончили цикл
end

В приведенном примере после запуска скрипта будет выведено:
CODE
loop 0.0000
loop 1.0000
loop 2.0000

Для того, чтобы организовать вложенные циклы, функции SaveIP и RestoreIP принимают дополнительный параметр (целочисленное значение), определяющий слот, в котором сохранена соответствующая метка. Если параметр не указан, по умолчанию принимается “0”. Всего во внутреннем списке имеется 256 слотов (последний номер - 255, поскольку нумерация начинается с “0”).
В языке СИ вложенные циклы можно организовать с помощью оператора For следующим образом:
CODE
for (i = 0; i < 5; i++)
{
for (j = 0; j < 3; j++)
{
// делаем что-то
}
}

Организовать подобное можно также с помощью функций OBSE:
CODE
short i
short j
Label 0; верхняя метка внешнего цикла (эквивалент SaveIP 0)
set j to 0
Label 1; верхняя метка внутреннего цикла
PrintToConsole "i = %.0f, j = %.0f" i, j
set j to (j + 1)
if (j < 3)
GoTo 1; эквивалент RestoreIP 1
endif
set i to (i + 1)
if (i < 5)
GoTo 0
endif

Отметьте, что если вышеприведенный пример не “завернут” в команду условия или в блок OnActivate, он будет запускаться во всех последующих фреймах.
Примечания:
На данный момент эта функция считается бета-версией и может работать ненадежно. Требуется дальнейшее тестирование.
Вопросы могут возникнуть, когда два или больше скриптов используют одни и те же коды меток.
Будьте очень осторожны и убедитесь, что ваш скрипт имеет корректное условие выхода из цикла, в противном случае вы запустите бесконечный цикл, который может заморозить игру и сделать невозможным нормальный выход из игры.
См. также: Label SaveIP RestoreIP
Относится к типу: OBSE Flow Control Functions

H

Con_HairTint

Синтаксис:
CODE
ActorRef.con_HairTint red green blue

Функция Con_HairTint изменяет цвет волос вызывающего актера ActorRef. Функция вызывается по указателю на объект.
Пример:
CODE
player.con_HairTint 0 0 0

В приведенном примере цвет волос персонажа игрока устанавливается в абсолютно черный цвет.
Примечания:
Из-за особенностей программного обеспечения Wiki название этой функции отображается некорректно – пропадает символ подчеркивания. Правильное название - Con_GetINISetting.
Эта функция идентична консольной команде GetINISetting и при вызове в скрипте ведет себя так, как будто Вы вызвали ее с консоли.
Эта функция не предназначалась для работы в скриптах, поэтому не будет работать так, как вам бы хотелось.
Относится к типу: OBSE Console Functions | OBSE Reference Functions

HammerKey

Синтаксис:
CODE
HammerKey Key

Функция HammerKey в каждом втором фрейме сообщает игре, что игрок как бы нажимает на указанную в виде параметра (Key) клавишу, хотя реального нажатия нет. Игра, получая эти сигналы, соответствующим образом реагирует. Функция HammerKey аналогична функции AHammerKey, за исключением того, что фреймы, в которых возвращаются значения этих функций, не совпадают (чередуются). Для отключения используйте функцию UnHammerKey. В качестве параметра принимаются стандартные сканкоды DirectX.
Примеры:
CODE
HammerKey 30; Симуляция нажатия на клавишу “A” (влево)
AHammerKey 32; Симуляция нажатия на клавишу “D” (вправо)

В приведенном примере игрок будет многократно дергаться влево-вправо (предполагается, что управление установлено по умолчанию).
Примечания:
Скан-коды DirectX обычно используют шестнадцатеричный код, но эта функция принимает десятичные значения.
Вы можете также задавать значения, большие 255, которые предназначены для управления мышью. При кодах от 256 до 263 вы можете работать с кнопками мыши. Коды 264 и 265 предназначены для колесика мыши.
Функции OBSE работают с аппаратно-независимыми кодами DirectX, в частности, модуля DirectInput. Перечень DX-сканкодов приведен в приложении 3: Сканкоды DX
См. также: AHammerKey, UnhammerKey, TapKey, HoldKey, ReleaseKey
Относится к типу: OBSE Input Functions

HoldKey

Синтаксис:
CODE
HoldKey Key

Функция HoldKey сообщает игре, что игрок удерживает указанную клавишу (Key) в нажатом состоянии, чтобы игра отреагировала соответствующим образом. Для отмены используйте функцию ReleaseKey. В качестве параметра принимаются стандартные сканкоды DirectX.
Примеры:
CODE
HoldKey 17; Симуляция постоянного нажатия на клавишу “W” (вперед)

В приведенном примере персонаж игрока будет постоянно передвигаться вперед, даже если на самом деле игрок ничего не нажимает на клавиатуре.
Примечания:
Скан-коды DirectX обычно используют шестнадцатеричный код, но эта функция принимает десятичные значения.

Вы можете также задавать значения, большие 255, которые предназначены для управления мышью. При кодах от 256 до 263 вы можете работать с кнопками мыши. Коды 264 и 265 предназначены для колесика мыши.
Функции OBSE работают с аппаратно-независимыми кодами DirectX, в частности, модуля DirectInput. Перечень DX-сканкодов приведен в приложении 3: Сканкоды DX
См. также: ReleaseKey, TapKey, HammerKey, AHammerKey, UnhammerKey
Относится к типу: OBSE Input Functions

I

IsActivator

Функция IsActivator может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Синтаксис:
CODE
IsActivator [ObjectID]

В данном случае функция IsActivator вызывается с указанием id объекта (ObjectID) в качестве параметра. Возвращает “1” (истину), если указанный объект является активатором. Этот вариант использования имеет приоритет перед вызовом на ссылке.
CODE
[Ref.]IsActivator

Здесь функция IsActivator вызывается с использованием указателя (Ref) на вызывающий объект и возвратит “1” (истину), если сам объект является активатором. Если скрипт “повешен” на самом объекте, то указатель на него указывать необязательно.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsAlchemyItem

Функция IsAlchemyItem может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Синтаксис:
CODE
IsAlchemyItem [ObjectID]

Возвращает “1” (истину), если объект классифицируется как зелье. Этот вариант использования имеет приоритет перед вызовом на ссылке.
CODE
[ref.]IsAlchemyItem

Пример вызова этой функции с помощью указателя на объект. В остальном работает также.
Если скрипт размещен на самом объекте, то указатель (ref) или параметр (ObjectID) указывать необязательно.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsAmmo

Функция IsAmmo может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Функция IsAmmo возвращает “1”, если вызываемый объект классифицируется как боеприпасы (в Обливионе к этому типу относятся только стрелы).
Синтаксис:
CODE
IsAmmo [ObjectID]

Вызов функции с явным указанием в виде параметра ObjectID. Этот вариант использования имеет приоритет перед вызовом с помощью указателя.
CODE
[ref.]IsAmmo

Пример вызова этой функции с помощью указателя ref.
Если скрипт повешен на сам объект, то указывать его ObjectID или ref необязательно.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsArmor

Функция IsArmor может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).

Функция IsArmor возвращает “1”, если вызывающий объект классифицируется как броня.
Синтаксис:
CODE
IsArmor [ObjectID]

Вызов функции с явным указанием ID объекта в виде параметра ObjectID. Этот вариант использования имеет приоритет перед вызовом с помощью указателя.
CODE
[ref.]IsArmor

Пример вызова этой функции с помощью указателя ref.
Если скрипт повешен на сам объект, то указывать его ObjectID или ref необязательно.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsBook

Функция IsArmor может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Функция IsBook возвращает “1”, если вызывающий объект классифицируется как книга.
Синтаксис:
CODE
IsBook [ObjectID]
Вызов функции с явным указанием ID объекта в виде параметра ObjectID. Этот вариант использования имеет приоритет перед вызовом с помощью указателя.
CODE
[ref.]IsBook

Пример вызова этой функции с помощью указателя ref.
Если скрипт повешен на сам объект, то указывать его ObjectID или ref необязательно.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsClothing

Функция IsClothing может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Функция IsClothing возвращает “1”, если вызывающий объект классифицируется как одежда.
Синтаксис:
CODE
IsClothing [ObjectID]

Вызов функции с явным указанием ID объекта в виде параметра ObjectID. Этот вариант использования имеет приоритет перед вызовом с помощью указателя.
CODE
[ref.]IsClothing

Пример вызова этой функции с помощью указателя ref.
Если скрипт повешен на сам объект, то указывать его ObjectID или ref необязательно.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsContainer

Функция IsContainer может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Функция IsContainer возвращает “1”, если вызывающий объект классифицируется как контейнер.
Синтаксис:
CODE
IsContainer [ObjectID]

Вызов функции с явным указанием ID объекта в виде параметра ObjectID. Этот вариант использования имеет приоритет перед вызовом с помощью указателя.
CODE
[ref.]IsContainer

Пример вызова этой функции с помощью указателя ref.
Если скрипт повешен на сам объект, то указывать его ObjectID или ref необязательно.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsDoor

Функция IsDoor может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Функция IsDoor возвращает “1”, если вызывающий объект классифицируется как дверь.
Синтаксис:
CODE
IsDoor [ObjectID]

Вызов функции с явным указанием ID объекта в виде параметра ObjectID. Этот вариант использования имеет приоритет перед вызовом с помощью указателя.
CODE
[ref.]IsDoor

Пример вызова этой функции с помощью указателя ref.
Если скрипт повешен на сам объект, то указывать его ObjectID или ref необязательно.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsFurniture

Функция IsFurniture может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Функция IsFurniture возвращает “1”, если вызывающий объект классифицируется как мебель.
Синтаксис:
CODE
IsFurniture [ObjectID]

Вызов функции с явным указанием ID объекта в виде параметра ObjectID. Этот вариант использования имеет приоритет перед вызовом с помощью указателя.
CODE
[ref.]IsFurniture

Пример вызова этой функции с помощью указателя ref.
Если скрипт повешен на сам объект, то указывать его ObjectID или ref необязательно.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsIngredient

Функция IsIngredient может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Функция IsIngredient возвращает “1”, если вызывающий объект классифицируется как ингредиент для алхимических опытов.
Синтаксис:
CODE
IsIngredient [ObjectID]

Вызов функции с явным указанием ID объекта в виде параметра ObjectID. Этот вариант использования имеет приоритет перед вызовом с помощью указателя.
CODE
[ref.]IsIngredient

Пример вызова этой функции с помощью указателя ref.
Если скрипт повешен на сам объект, то указывать его ObjectID или ref необязательно.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsKey

Функция IsKey может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Функция IsKey возвращает “1”, если вызывающий объект классифицируется как ключ (игровой).
Синтаксис:
CODE
IsKey [ObjectID]

Вызов функции с явным указанием ID объекта в виде параметра ObjectID.
CODE
[ref.]IsKey

Пример вызова этой функции с помощью указателя ref.
Если скрипт повешен на сам объект, то указывать его ObjectID или ref необязательно.

Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsKeyPressed

Синтаксис:
CODE
IsKeyPressed Key

Функция IsKeyPressed возвращает “1”, если указанная в виде параметра (Key) клавиша в данный момент нажата. Код клавиши хранится в заголовочных файлах Windows, в отличие от функции IsKeyPressed2, которая использует аппаратно независимые сканкоды DirectX.
Некоторые Windows-сканкоды клавиш (Key IDs)
CODE
1 Left Mouse Button
2 Right Mouse Button
4 Middle Mouse Button
---
13 Enter
---
20 Caps Lock
---
32 spacebar
---
33 PgUp
34 PgDn
35 End
36 Home
37 Left
38 Up
39 Right
40 Down
44 PrtScr
45 Ins
46 Del
---
48 0
49 1
50 2
...
55 7
56 8
57 9
---
65 A
66 B
67 C
...
88 X
89 Y
90 Z
---
96 NUM0
97 NUM1
98 NUM2
...
103 NUM7
104 NUM8
105 NUM9
106 NUM*
107 NUM+
109 NUM-
111 NUM/
---
112 F1
113 F2
114 F3
...
123 F12

...
127 F16
---
160 left shift
161 right shift
162 left control
163 right control

Полный список скан-кодов вы можете найти в приложении 5.

Примечания:
Эта функция не будет возвращать значение, если отображается сообщение MessageBox, возможно потому, что ввод переключается на его окно.
Помните, что эта функция возвращает 1 пока клавиша нажата. Так что, это хороший способ, чтобы подождать, пока её не отпустят:
CODE
if (curKey && isKeyPressed curkey); клавиша остаётся нажатой
Return; ждём отпускания клавиши
else
set curkey to 0
endif
if (isKeyPressed <keyCode>)
; делаем что-либо
set curKey to <keyCode>
endif
См. также: IsKeyPressed2
Относится к типу: OBSE Input Functions

IsKeyPressed2

Синтаксис:
CODE
IsKeyPressed2 Key

Функция IsKeyPressed2 возвращает “1”, если указанная клавиша Key в данный момент нажата. В качестве параметра IsKeyPressed2 принимает аппаратно-независимые сканкоды DirectX, в отличие от IsKeyPressed, которая работает с виртуальными сканкодами Windows.
Примечания:
Сканкоды DirectX обычно работают с числами в шестнадцатеричном исчислении, однако функция IsKeyPressed2 принимает десятичные значения
Вы можете также задавать значения, большие 255, которые предназначены для управления мышью. При кодах от 256 до 263 вы можете работать с кнопками мыши. Коды 264 и 265 предназначены для колесика мыши.
Функция работает с аппаратно-независимыми кодами DirectX, в частности, модуля DirectInput. Перечень DX-сканкодов приведен в приложении 3: Сканкоды DX
Помните, что эта функция будет возвращать “1” до тех пор, пока клавиша нажата, так что это хороший способ, чтобы перехватить нажатие и затем подождать, пока её не отпустят:
CODE
if (curKey && isKeyPressed2 curkey); клавиша всё ещё нажата
return; ждём, пока клавишу отпустят
else
set curkey to 0
endif

if (isKeyPressed2 <keyCode>)
; делаем что-нибудь
set curKey to <keyCode>
endif
См. также: IsKeyPressed
Относится к типу: OBSE Input Functions

IsLight

Синтаксис:
CODE
IsLight [ObjectID]

Функция IsLight возвращает “1”, если указанный в виде параметра ObjectID объект относится к источнику света.
CODE

[ref.]IsLight

Функция IsLight возвращает “1”, если вызывающий по указателю (ref) объект относится к источнику света.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsPotion

Функция IsPotion может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Синтаксис:
CODE
IsPotion [ObjectID]

Возвращает “1” (истину), если объект классифицируется как яд. Этот вариант использования имеет приоритет перед вызовом на ссылке.
CODE
[ref.]IsPotion

Здесь функция вызывается с помощью указателя на объект и возвращает “1”, если этот объект классифицируется как яд. Отметьте, что яд – это алхимический предмет, в котором все эффекты отрицательные. Если имеется хоть один неотрицательный эффект, то такой алхимический предмет ядом не считается и относится к зельям.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsRefEssential

Синтаксис:
CODE
[Ref.]IsRefEssential

Вызывается с помощью указателя на персонаж. Возвращает “1”, если данный персонаж является важным для игры (для квестов) и не может быть убит.
Пример:
CODE
if (myNPCRef.isEssential)
myNPCRef.setRefEssential 0; удаляем флаг essential с персонажа
endif

См. также: SetRefEssential
Относится к типу: OBSE Reference Functions | OBSE Actor Functions

IsSoulGem

Синтаксис:
CODE
IsSoulGem [ObjectID]

Функция IsSoulGem возвращает “1”, если указанный объект является камнем душ.
CODE
[ref.]IsSoulGem

Функция IsSoulGem возвращает “1”, если объект, указанный с помощью указателя (ref), является камнем душ.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

IsWeapon

Синтаксис:
CODE
IsWeapon [ObjectID]

Функция IsWeapon возвращает “1”, если указанный в виде параметра объект классифицируется как оружие.
CODE
[ref.]IsWeapon

Функция IsWeapon возвращает “1”, если объект, указанный с помощью указателя (ref), классифицируется как оружие.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

L

Label

Синтаксис:
CODE
Label [labelID]

Функция Label позволяет установить несколько меток (labelID) внутри скрипта. Она идентична команде SaveIP. Использование команд GoTo или RestoreIP вызовет переход на метку, что позволяет организовывать простые циклы. Благодаря опциональному флагу Label ID можно установить несколько меток или создать вложенные циклы.
Пример:
CODE
begin onActivate
; объявление переменной, в которой будет храниться количество итераций цикла
long var
set var to 0

; сохранение указателя на текущую инструкцию
; эти записи в начале цикла должны находиться до команды PrintToConsole
SaveIP; также можно написать Label

; вывод номера текущей итерации
PrintToConsole "loop %f" var

; обновление счетчика
set var to var + 1

; мы хотим выполнить цикл только три раза; проверка значения счетчика
if var < 3
; переход назад, к сохранённой позиции
RestoreIP; could also use Goto
endif

; если мы попали сюда, то цикл закончился
end

После активации в игре этот скрипт выведет на экран следующие сообщения:
CODE
loop 0.0000
loop 1.0000
loop 2.0000

Для поддержки вложенных циклов команды SaveIP и RestoreIP могут принимать необязательный целочисленный параметр, указывающий "слот" для сохранения адреса инструкции. По умолчанию он равен нулю. Всего доступно 256 слотов.

Пример, более похожий на цикл for из нормального языка программирования, вроде этого:
CODE
for (i = 0; i < 5; i++)
{
for (j = 0; j < 3; j++)
{
//Делаем что-либо
}
}

Создание подобного цикла с использованием функций OBSE:
CODE
short i
short j

Label 0; top of outer loop, equivalent to SaveIP 0
set j to 0
Label 1; top of inner loop
PrintToConsole "i = %.0f, j = %.0f" i, j
set j to (j + 1)
if (j < 3)
GoTo 1; equivalent to RestoreIP 1
endif
set i to (i + 1)

if (i < 5)
GoTo 0
endif

Отметьте, что если вышеприведенный пример не “завернут” в команду условия или в блок OnActivate, он будет запускаться во всех последующих фреймах.
Примечание:
Функции находятся в стадии бета-тестирования и могут быть ненадёжны.
Могут появится конфликты когда два или более скриптов используют одни и те же коды меток.
Убедитесь что ваш цикл имеет условие выхода, иначе вы рискуете создать бесконечный цикл, который подвесит игру и не даст возможности игроку выйти из неё.
См. также: SaveIP, RestoreIP, GoTo
Относится к типу: OBSE Flow Control Functions

Log

Синтаксис:
CODE
Log [float]

Функция Log возвращает натуральный логарифм числа.
Пример:
CODE
set n to log n

См. также: log10, exp
Относится к типу: OBSE Math Functions

Log10

Синтаксис:
CODE
log10 [float]

Функция Log10 возвращает логарифм числа по основанию 10.
Пример:
CODE
set n to log10 n

См. также: log, exp
Относится к типу: OBSE Math Functions

M

ModActorValue2

Синтаксис:
CODE
ModActorValue2 StatName value:integer
ModAV2 StatName value:integer

Пример:
CODE
ModActorValue2 Strength -10

Функция ModActorValue2 изменяет текущее значение характеристики на указаную величину, не изменяя при этом её базовое значение. Value - это целочисленная, положительная или отрицательная величина. ModAV2 StatName 3.9 приведёт к изменению на 3 единицы.

Попытка задать исключительно положительным характеристикам (например, здоровья) отрицательное значение приведёт к вылету из игры.

В отличие от ModActorValue, эта функция работает как заклинание:
Отрицательные значение могут быть удалены заклинанием восстановления.
Положительные значения восстановят ущерб, нанесённый характеристике, но характеристика не может превзойти своего максимального значения.
("Player.ModAV2 health 10000" полностью вылечит, но не даст вам 10000 здоровья).
Примечания: Эта функция идентична консольной версии ModActorValue

Относится к типу: OBSE Item Functions | OBSE Reference Functions

Con_ModWaterShader

Синтаксис:
CODE
Con_ModWaterShader string float

Аналог - MWS.
Примечание:
Из-за ограничений Wiki, название данной статьи и ссылки на неё не отображаются корректно - в названии функции есть подчеркивание (_).
Эта функция является эквивалентом консольной команды ModWaterShader и будет работать так, как будто была вызвана из консоли.
Эта функция не расчитана на использование в скриптах, поэтому может вести себя неожиданно.
Относится к типу: OBSE Console Functions

MoveMouseX

Синтаксис:
CODE
MoveMouseX pixels

Функция MoveMouseX сообщает игре, что мышь была сдвинута на определённое число пикселей по горизонтали. Игра отреагирует в зависимости от ситуации.
Примеры:
CODE
MoveMouseX -50

В приведенном примере курсор мыши будет перемещен на 50 пикселей левее.
См. также: MoveMouseY, SetMouseSpeedX, SetMouseSpeedY, DisableMouse, EnableMouse
Относится к типу: OBSE Input Functions

MoveMouseY

Синтаксис:
CODE
MoveMouseY pixels[/cpde]
Функция MoveMouseY сообщает игре, что мышь была сдвинута на определённое число пикселей по вертикали. Игра отреагирует в зависимости от ситуации.
Пример:
CODE
MoveMouseY -50

Сдвигает мышь на 50 пикселей вниз или вверх в зависимости от того, включено ли инвертирование мыши.
См. также: MoveMouseX, SetMouseSpeedX, SetMouseSpeedY, DisableMouse, EnableMouse
Относится к типу: OBSE Input Functions

P

Pow

Синтаксис:
CODE
Pow [base] [exponent]

Функция Pow возвращает основание (base), возведенное в указанную степень (exponent). Все значения имеют тип float (вещественные).
Пример:
CODE
set n to (pow x y)

Относится к типу: OBSE Math Functions

PrintToConsole

Синтаксис:
CODE

PrintToConsole "Message text", [var1], [var2], etc

Функция PrintToConsole выводит форматированные строки в консоли. Используется такой же синтаксис, как и при вызове функций Message и MessageBox. Большей частью полезны для целевой отладки.
Относится к типу: OBSE Debug Functions

PurgeCellBuffers

Синтаксис:
CODE
PurgeCellBuffers

Функция PurgeCellBuffers удаляет из памяти текстуры или модели, которые находятся вне загруженной в данный момент области. Использование этой функци очистит память и улучшит быстродействие (fps) игры.

Данная возможность была добавлена первым официальным патчем (v1.1.511) в Oblivion.exe, но не в TESConstructionSet.exe. Её можно вызывать из консоли, однако редактор скриптов не признаёт её корректной и никогда не скомпилирует скрипт, содержащий данную функцию.

Oblivion Script Extender позволяет скриптерам использовать данную функцию, но, в отличие от остальных функций OBSE, её использование не требует наличия OBSE. Плагины, у которых в скриптах имеется эта скомпилированная фунция, будут работать также и у тех, у кого OBSE не установлен.

В отличие от других функций OBSE, которые только иммитируют консольные функции, эта функция была создана для использования в скриптах и теперь это возможно.
Относится к типу: OBSE Console Functions

R

Rand

Синтаксис:
CODE
rand [min] [max]

Функция Rand возвращает случайно сгенерированное число, которое находится в диапазоне от min до max и указанных в виде параметров.
Пример:
CODE
set n to rand 1 100

См. также: GetRandomPercent
Относится к типу: OBSE Math Functions

Con_RefreshINI

Синтаксис:
CODE
con_RefreshINI

Функция Con_RefreshINI используется для сброса в значение по умолчанию измененных параметров INI-файла. Изменения могли быть произведены функцией SetINISetting.
Пример:
CODE
con_RefreshINI

Используется после вызова функции SetINISetting.
Примечание:
Из-за особенностей программного обеспечения Wiki название этой функции отображается некорректно – пропадает символ подчеркивания. Правильное название - Con_RefreshINI.
Эта функция идентична консольной команде RefreshINI и при вызове в скрипте ведет себя так, как будто Вы вызвали ее с консоли.
Эта функция не предназначалась для работы в скриптах, поэтому не будет работать так, как вам бы хотелось.
Относится к типу: OBSE Console Functions

ReleaseKey

Синтаксис:

CODE
ReleaseKey key

Функция ReleaseKey отменяет действия, вызванные функцией HoldKey. Эта функция использует сканкоды DirectX.
Пример:
CODE
if (timer == 0)
HoldKey 17; Клавиша “W”
elseif (timer < 30)
set timer to (timer + GetSecondsPassed)
else
ReleaseKey 17
endif

В приведенном примере персонаж игрока будет идти вперед 30 секунд, затем остановится (подразумевается, что клавиши управления установлены по умолчанию).
Примечания:
Скан-коды DirectX обычно используются в шестнадцатеричном коде, но эта функция принимает числа в десятичном исчислении.
Вы можете также задавать значения, большие 255, которые предназначены для управления мышью. При кодах от 256 до 263 вы можете работать с кнопками мыши. Коды 264 и 265 предназначены для колесика мыши.
Перечень сканкодов смотрите в приложении 3: Сканкоды DX
См. также: HoldKey, TapKey, HammerKey, AHammerKey, UnhammerKey
Относится к типу: OBSE Input Functions

RestoreIP

Синтаксис:
CODE
RestoreIP [labelID]

Функция RestoreIP позволяет в процессе выполнения скрипта перейти на метку labelID, определенную в пределах этого же скрипта. Работает аналогично функции GoTo. Для установки метки вы можете использовать функцию SaveIP (для функции GoTo – с помощью метки Label). Явное указание необязательного параметра labelID может использоваться при использовании нескольких меток в пределах скрипта или организовать вложенные ветвящиеся циклы.
Пример:
CODE
begin onActivate
; объявляем переменную, в которой будем отслеживать количество выполненных циклов
long var
set var to 0
; сохраняем текущие установки указателя
; записываем начало цикла, который находится непосредственно перед командой PrintToConsole
SaveIP; устанавливаем метку, можно также было использовать Label.
; выводим текущий отсчет цикла
PrintToConsole "loop %f" var
; обновляем счетчик цикла
set var to var + 1
; нам нужно выполнить цикл всего три раза, поэтому проверяем состояние счетчика
if var < 3
; переходим на сохраненную метку
RestoreIP; здесь можно использовать функцию GoTo, если ранее использовалась Label
endif
; если мы дошли в это место, значит мы закончили цикл
End

В приведенном примере после запуска скрипта будет выведено:
CODE
loop 0.0000
loop 1.0000
loop 2.0000

Для того, чтобы организовать вложенные циклы, функции SaveIP и RestoreIP принимают дополнительный параметр (целочисленное значение), определяющий слот, в котором сохранена соответствующая метка. Если параметр не указан, по умолчанию принимается “0”. Всего во внутреннем списке имеется 256 слотов (последний номер - 255, поскольку нумерация начинается с “0”).
В языке СИ вложенные циклы можно организовать с помощью оператора For следующим образом:
CODE

```
for (i = 0; i < 5; i++)
{
for (j = 0; j < 3; j++)
{
//делаем что-нибудь
}
}
```

Организовать подобное можно также с помощью функций OBSE:

```
CODE
short i
short j
Label 0; верхняя метка внешнего цикла (эквивалент SaveIP 0)
set j to 0
Label 1; верхняя метка внутреннего цикла
PrintToConsole "i = %.0f, j = %.0f" i, j
set j to ( j + 1 )
if ( j < 3 )
GoTo 1; эквивалент RestoreIP 1
endif
set i to ( i + 1 )
if ( i < 5 )
GoTo 0
endif
```

Отметьте, что если вышеприведенный пример не “завернут” в команду условия или в блок OnActivate, он будет запускаться во всех последующих фреймах.
Примечания:
На данный момент эта функция считается бета-версией и может работать ненадежно. Требуется дальнейшее тестирование.
Вопросы могут возникнуть, когда два или больше скриптов используют одни и те же коды меток.
Будьте очень осторожны и убедитесь, что ваш скрипт имеет корректное условие выхода из цикла, в противном случае вы запустите бесконечный цикл, который может заморозить игру и сделать невозможным нормальный выход из игры.
См. также: SaveIP, Label, GoTo
Относится к типу: OBSE Flow Control Functions

Con_RunMemoryPass

Синтаксис:

```
CODE
con_RunMemoryPass Flag
```

Функция con_RunMemoryPass запускает проход очистки памяти.

Примечания:
Из-за ограничений Wiki, название данной статьи и ссылки на неё не отображаются корректно - в названии функции есть подчеркивание (_).
Эта функция является эквивалентом консольной функции RunMemoryPass и будет работать так, как будто была вызвана из консоли.
Эта функция не расчитана на использование в скриптах и может вести себя неожиданно.
Относится к типу: OBSE Console Functions

S

SaveIP

Синтаксис:

```
CODE
SaveIP [labelID]
```

Функция SaveIP позволяет установить несколько меток (labelID) внутри скрипта. Эта функция идентична функции Label. Использование команд RestoreIP (или GoTo, если метка устанавливалась с помощью функции Label) вызовет переход на установленную метку, что позволяет организовывать простые циклы. Благодаря опциональному флагу (LabelID) можно установить несколько меток или создавать вложенные циклы.
Примеры:

```
CODE
begin onActivate
; объявление переменной, в которой будет храниться количество итераций цикла
long var
```

```
set var to 0

; сохранение указателя на текущую инструкцию
; эти записи в начале цикла должны находиться до команды PrintToConsole
SaveIP; также можно написать Label

; вывод номера текущей итерации
PrintToConsole "loop %f" var

; обновление счетчика
set var to var + 1

; мы хотим выполнить цикл только три раза; проверка значения счетчика
if var < 3
; переход назад, к сохранённой позиции
RestoreIP; could also use Goto
endif

; если мы попали сюда, то цикл закончился
end
```

После активации в игре этот скрипт выведет на экран следующие сообщения:

```
CODE
loop 0.0000
loop 1.0000
loop 2.0000
```

Для того, чтобы организовать вложенные циклы, функции SaveIP и RestoreIP принимают дополнительный параметр (целочисленное значение), определяющий слот, в котором сохранена соответствующая метка. Если параметр не указан, по умолчанию принимается “0”. Всего во внутреннем списке имеется 256 слотов (последний номер - 255, поскольку нумерация начинается с “0”).
В языке СИ вложенные циклы можно организовать с помощью оператора For следующим образом:

```
CODE
for (i = 0; i < 5; i++)
{
for (j = 0; j < 3; j++)
{
// делаем что-то
}
}
```

Организовать подобное можно также с помощью функций OBSE:

```
CODE
short i
short j
Label 0; верхняя метка внешнего цикла (эквивалент SaveIP 0)
set j to 0
Label 1; верхняя метка внутреннего цикла
PrintToConsole "i = %.0f, j = %.0f" i, j
set j to ( j + 1 )
if ( j < 3 )
GoTo 1; эквивалент RestoreIP 1
endif
set i to ( i + 1 )
if ( i < 5 )
GoTo 0
endif
```

Отметьте, что если вышеприведенный пример не “завернут” в команду условия или в блок OnActivate, он будет запускаться во всех последующих фреймах.
Примечания:
На данный момент эта функция считается бета-версией и может работать ненадежно. Требуется дальнейшее тестирование.
Вопросы могут возникнуть, когда два или больше скриптов используют одни и те же коды меток.
Будьте очень осторожны и убедитесь, что ваш скрипт имеет корректное условие выхода из цикла, в противном случае вы запустите бесконечный цикл, который может заморозить игру и сделать невозможным нормальный выход из игры.
См. также: Label, RestoreIP, GoTo
Относится к типу: OBSE Flow Control Functions

SetModelPath

Синтаксис:
CODE
[Ref.]SetModelPath "path string" [objectID]

Функция SetModelPath устанавливает модель для базового объекта. Может вызываться двумя различными путями. В первом случае объект указывается в виде необязательного параметра (ObjectID), а во втором – как указатель на объект (ref).
Первый параметр ("path string") задает путь к файлу *.nif с новой моделью.
Путь к файлу должен содержать Oblivion\Data\Meshes.
Если файл требуемый *.nif не найден, то к объекту не будет подключено никакой модели.
Эта функция работает только с простыми объектами, с которыми связана только одна модель, как-то активаторы, статика, большинство объектов инвентаря и т.п. Функция не затрагивает одежду и броню (для этого используйте SetMaleBipedPath и подобные). Если вызвать эту функцию на актере, то позже это может привести к вылету из игры, когда актер будет перезагружен или удален.
Эта функция изменяет модель для базового объекта, а это означает, что и все копии объекта того же типа, которые впоследствии будут загружены, также “приобретут” новую модель.
Все изменения не сохраняются в “сохраненках” (за исключением клонированных моделей (форм)), поэтому они после перезагрузки игры снова приобретут первоначальный вид.
Новая модель не будет отображаться до тех пор, пока объект не будет перезагружен. Это означает, например, что те объекты, которые могут быть экипированы, должны быть сняты, а затем надеты обратно. Что касается объектов игрового мира, то вызов Disable после Enable тут же обновит модель на новую.
Относится к типу: OBSE Item Functions | OBSE Reference Functions

SetMouseSpeedX

Синтаксис:
CODE
SetMouseSpeedX speed

Функция SetMouseSpeedX перемещает мышь в горизонтальном направлении со скоростью, равной указанному в виде параметра числу пикселей за секунду (speed).
Примеры:
CODE
SetMouseSpeedX 5

В приведенном примере курсор мыши будет передвигаться со скоростью 5 пикселей в секунду вправо.
См. также: SetMouseSpeedY, MoveMouseX, MoveMouseX, DisableMouse, EnableMouse
Относится к типу: OBSE Input Functions

SetMouseSpeedY

Синтаксис:
CODE
SetMouseSpeedY speed

Функция SetMouseSpeedY перемещает мышь в вертикальном направлении со скоростью, равной указанному в виде параметра числу пикселей за секунду (speed).
Примеры:
CODE
SetMouseSpeedY 5

В приведенном примере курсор мыши будет передвигаться со скоростью 5 пикселей в секунду вверх.
См. также: SetMouseSpeedX, MoveMouseX, MoveMouseX, DisableMouse, EnableMouse
Относится к типу: OBSE Input Functions

SetRefEssential

Синтаксис:
CODE
[Ref.]setRefEssential booleanFlag

Функция setRefEssential должна вызываться на указателе на персонажа (ref). Установка флага booleanFlag в “1” позволит пометить актера как “Essential” - важного для игры персонажа, которого нельзя убить. Данный флаг вы могли видеть в конструкторе – он есть у всех персонажей – как NPC, так и существ – Creature. Обычно это необходимо для того, чтобы игрок случайно либо намеренно не убил очень важного для прохождения квеста актера. Если его здоровье упадет до 0, он просто потеряет на некоторое время сознание. Повторный вызов этой функции с параметром “0” сбросит этот флаг, и его снова можно будет убить.

Пример:
CODE
if (myNPCRef.isEssential == 0)
myNPCRef.setEssential 1; устанавливаем на персонаже в игре флаг “неубиваемый”
endif

См. также: IsRefEssential
Относится к типу: OBSE Actor Functions | OBSE Reference Functions

Con_SetCameraFOV

Синтаксис:
CODE
con_SetCameraFOV Degrees

Функция con_SetCameraFOV устанавливает угол зрения камеры в указанное значение (Degrees). Единица измерения – градусы, по умолчанию равен 75°.
Примечания:
Из-за ограничений синтаксиса Wiki название данной статьи и ссылки на неё отображаются не совсем корректно - в названии функции должен стоять символ подчеркивания (“_”), а не пробел.
Эта функция является эквивалентом консольной команды SetCameraFOV и будет работать так, как будто была вызвана из консоли.
Эта функция не рассчитана на использование в скриптах, поэтому может вести себя неожиданно.
Относится к типу: OBSE Console Functions

Con_SetClipDist

Синтаксис:
CODE
con_SetClipDist ClipDistance

Функция con_SetClipDist с помощью параметра ClipDistance устанавливает расстояние (в игровых единицах длины), на котором перестают определяться коллизии (Collision Detection).
Краткая информация: Напомним, что коллизии – это пересечение множества точек, принадлежащих к различным трехмерным телам в процессе их соприкосновения. Факт столкновения обычно проверяется, чтобы объекты вели себя корректно, с учетом физики, заложенной в игровой мир (Rigid Body Dynamics). Чем больше взаимодействующих объектов находится в заданном от камеры радиусе, тем ниже будет быстродействие. Точное определение пересечений требует полного пересчета всех треугольников моделей, что также довольно существенно снижает fps. Поэтому на практике используются различные оптимизированные алгоритмы.
Пример:
CODE
con_SetClipDist 4096

В приведенном примере дистанция установлена равной 4096 игровых единиц длины, что равно длине одной стороны квадрата внешней игровой ячейки.
Примечания:
Из-за ограничений синтаксиса Wiki название данной статьи и ссылки на неё отображаются не совсем корректно - в названии функции должен стоять символ подчеркивания (“_”), а не пробел.
Эта функция является эквивалентом консольной команды SetClipDist и будет работать так, как будто была вызвана из консоли.
Эта функция не рассчитана на использование в скриптах, поэтому может вести себя неожиданно.
Относится к типу: OBSE Console Functions

Con_SetFog

Синтаксис:
CODE
con_SetFog StartDistance EndDistance

Функция con_SetFog устанавливает глубину тумана в игре (в игровых единицах длины). Начальное значение задается параметром StartDistance, а конечное – EndDistance.
Пример:
CODE
con_SetFog 2000.0 5000.0

В приведенном примере туман установлен на стартовое значение в 2000 единиц, а конечное – в 5000.
Примечания:

Из-за ограничений синтаксиса название данной статьи на официальном Wiki отображаются не совсем корректно - в названии функции должен стоять символ подчеркивания (“_”), а не пробел.
Эта функция является эквивалентом консольной команды SetFog и будет работать так, как будто была вызвана из консоли.
Эта функция не рассчитана на использование в скриптах, поэтому может вести себя неожиданно.
Относится к типу: OBSE Console Functions

Con_SetGameSetting

Синтаксис:
CODE
con_SetGameSetting GMST value

Функция Con_SetGameSetting устанавливает указанное значение игровой установки GMST в требуемое значение value.
Пример:
CODE
con_SetGameSetting iMagicMaxSummonedCreatureTypes 10

В приведенном примере накладывается предел на общее количество одновременно вызываемых существ (не выше 10).
Примечания:
Из-за ограничений синтаксиса название данной статьи на официальном Wiki отображаются не совсем корректно - в названии функции должен стоять символ подчеркивания (“_”), а не пробел.
Эта функция является эквивалентом консольной команды SetGameSetting и будет работать так, как будто была вызвана из консоли.
Эта функция не рассчитана на использование в скриптах, поэтому может вести себя неожиданно.
Con_SetGameSetting работает некорректно, если в качестве параметра указать переменную. Если вам все же нужно использовать переменную, то используйте функцию SetNumericGameSetting.
Изменения, сделанные этой функцией в игровых установках, не будут сохранены в “сохраненке”. Отсюда вытекают 2 вещи:
Если Вы хотите сделать изменение в игровых установках постоянным, Вам нужно сделать так, чтобы скриптовые коды обновляли величину установки всякий раз, как будет перезагружено сохранение.
Когда вы изменяете игровые установки, а затем перезагружаете сохранение, изменение все еще будет активно. Игровые установки сбросятся только тогда, когда вы полностью выйдете из игры и зайдете в нее заново, или когда после перезагрузки сохранения вы сделаете это в вашем скрипте.

См. также: SetNumericGameSetting, GetGameSetting
Относится к типу: OBSE Console Functions

SetNumericGameSetting

Синтаксис:
CODE
SetNumericGameSetting GMST value

Функция SetNumericGameSetting устанавливает в игре значение игровой настройки GMST в требуемое значение (Value).
Пример:
CODE
SetNumericGameSetting iMagicMaxSummonedCreatureTypes 10
SetNumericGameSetting iMagicMaxSummonedCreatureTypes Variable

В приведенном примере накладывается предел на общее количество одновременно вызываемых существ (не выше 10).
Примечания:
Эта функция идентична Con_SetGameSetting с той разницей, что может принимать не только числовые значения, но и переменные.
Изменения, сделанные этой функцией в игровых установках, не будут сохранены в “сохраненке”. Отсюда вытекают 2 вещи:
Если Вы хотите сделать изменение в игровых установках постоянным, Вам нужно сделать так, чтобы скриптовые коды обновляли величину установки всякий раз, как будет перезагружено сохранение.
Когда вы изменяете игровые установки, а затем перезагружаете сохранение, изменение все еще будет активно. Игровые установки сбросятся только тогда, когда вы полностью выйдете из игры и зайдете в нее заново, или когда после перезагрузки сохранения вы сделаете это в вашем скрипте.

См. также: Con_SetGameSetting, GetGameSetting
Относится к типу: OBSE Console Functions

Con SetGamma

Синтаксис:
CODE
con_SetGamma float

Функция con_SetGamma Sets устанавливает новые значения таблиц гамма-коррекции (gamma ramp).
Garin: Справка.
Гамма-коррекция (gamma correction). В графической библиотеке GL существует специальная процедура для таблиц задания интенсивности красного, синего и зеленого цветов, которые изменяются по логарифмическому закону. Связано это с особенностями восприятия человеческого глаза интенсивностей освещенных сцен, которые гораздо ближе к логарифмическому закону, чем к линейному. Для каждого из трех цветов имеется своя таблица преобразования. Эти таблицы в совокупности получили название “Таблицы гамма-коррекции” (gamma ramp, не путайте с обычными таблицами цветов).
Таблицы гамма-коррекции управляют работой электронной пушки монитора по логарифмическому закону. Записи в таблицах гамма-коррекции сглаживают различия между люминесцентными поверхностями мониторов различных производителей за счет нелинейной характеристики и широкого динамического диапазона.
Прим. Zomb: При тестировании обнаружилось, что изменяется яркость картинки. От "всё белым-бело" при 0,01 до "и пришла тьма" при 1000. Но пределов, кажется, нет. Стандартное значение у меня было где-то чуть меньше 1.
Примечания:
Из-за ограничений синтаксиса название данной статьи на официальном Wiki отображаются не совсем корректно - в названии функции должен стоять символ подчеркивания (“_”), а не пробел.
Эта функция является эквивалентом консольной команды SetGamma (краткое название SG) и будет работать так, как будто была вызвана из консоли.
Эта функция не рассчитана на использование в скриптах, поэтому может вести себя неожиданно.
Относится к типу: OBSE Console Functions

Con_SetHDRParam

Синтаксис:
CODE
con_SetHDRParam EyeAdaptSpeed EmissiveHDRMult TreeDimmer GrassDimmer value5 value6

EyeAdaptSpeed, fEmissiveHDRMult, fTreeDimmer, fGrassDimmer, fValue5, fValue6
Функция con_SetHDRParam устанавливает различные параметры для шейдеров HDR (HDR shader).
Все параметры – вещественные переменные.
Garin: Справка.
HDR – аббревиатура “High Dynamic Range”, что можно перевести как “широкий динамический диапазон”. HDR - новомодная технология, позволяющая получить реалистичные высококачественные картинки с использованием “третьих” шейдеров. Настройки HDR можно встретить как в конструкторе, так и в файлах *.ini (есть пользовательский ini-файл с настройками, и есть по умолчанию.) В КС настройки находятся, например, в меню World\Weather и позволяют настроить HDR для каждого типа погоды. Во время смены погоды настройки HDR смешиваются. Если у пользователя отключен HDR, эти настройки игнорируются. Если все значения равны нулю, то настройки будут взяты из ini-файла.
Параметры, которые задаются в функции con_SetHDRParam, имеют свои аналоги на закладке HDR в конструкторе:
EyeAdaptSpeed – настройка Eye Adapt Speed (скорость адаптации зрения, вещественное число в интервале от 0 до 1) влияет на то, как долго камера будет адаптироваться к новому уровню освещённости. Чем меньше значение, тем быстрее адаптация.
EmissiveHDRMult – настройка Emissive Mult (вещественное, положительное число) - чем больше значение - тем ярче объекты с emissive lighting.
TreeDimmer - настройка Tree Dimmer (вещественное число) - устанавливает, насколько ярко выглядят деревья. Чем больше значение - тем ярче.
GrassDimmer – настройка Grass Dimmer (вещественное число) - Устанавливает, насколько ярко выглядит трава. Чем больше значение - тем она ярче.
value5 – требуется тестирование.
value6 – требуется тестирование.
Примечания:
Из-за ограничений синтаксиса название данной статьи на официальном Wiki отображаются не совсем корректно - в названии функции должен стоять символ подчеркивания (“_”), а не пробел.
Эта функция является эквивалентом консольной команды SetHDRParam (краткое название SHP) и будет работать так, как будто была вызвана из консоли.
Эта функция не рассчитана на использование в скриптах, поэтому может вести себя неожиданно.
Относится к типу: OBSE Console Functions

Con_SetImageSpaceGlow

Синтаксис:
CODE
con_SetImageSpaceGlow int1 int2 int3 int4

Функция con_SetImageSpaceGlow - аналог консольной функции SISG. Примерный перевод названия - “установить космическое свечение изображения”. Принимает 4 целочисленных параметра int1…int4.
Прим. Zig: Великолепный спецэффект. Задайте параметры функции 1 2 3 4 (или наберите в консоли sisg 1 2 3 4) и Обливион станет заметно веселей.
Примечания:
Требуется дополнительное тестирование – не известно, какие параметры за что отвечают.

Из-за ограничений синтаксиса название данной функции на официальном Wiki отображается не совсем корректно - в названии функции должен стоять символ подчеркивания (“_”), а не пробел.
Эта функция является эквивалентом SetImageSpaceGlow Console Function (краткое название SISG), и будет работать так, как будто была вызвана из консоли.
Эта функция не рассчитана на использование в скриптах, поэтому может вести себя неожиданно.
Относится к типу: OBSE Console Functions

Con_SetINISetting

Синтаксис:
CODE
con_SetINISetting Setting value

Функция устанавливает указанный параметр (Setting) в Oblivion.ini в нужное значение (value).
Пример:
CODE
con_SetINISetting bCrossHair 0

В приведенном примере функция выключает курсор прицела (crosshair).
Примечания:
Из-за ограничений синтаксиса название данной функции на официальном Wiki отображается не совсем корректно - в названии функции должен стоять символ подчеркивания (“_”), а не пробел.
Эта функция является эквивалентом консольной команды SetINISetting (краткое название SetINI) и будет работать так, как будто была вызвана из консоли.
Эта функция не рассчитана на использование в скриптах, поэтому может вести себя неожиданно.
Относится к типу: OBSE Console Functions

Con_SetTargetRefraction

Синтаксис:
CODE
actorRef.con_SetTargetRefraction value

Функция con_SetTargetRefraction устанавливает значение рефракции (refractive) для вызывающего актера (устанавливает преломляющую способность цели).
Примечания:
Из-за ограничений синтаксиса название данной функции на официальном Wiki отображается не совсем корректно - в названии функции должен стоять символ подчеркивания (“_”), а не пробел.
Эта функция является эквивалентом консольной команды SetTargetRefraction (краткое название STR) и будет работать так, как будто была вызвана из консоли.
Эта функция не была изначально рассчитана на использование в скриптах, поэтому может вести себя неожиданно.
См. также: SetTargetRefractionFire
Относится к типу: OBSE Console Functions

Con SetTargetRefractionFire

Синтаксис:
CODE
actorRef.con_SetTargetRefraction value1 value2

Функция con_SetTargetRefraction устанавливает преломляющую способность огня для вызывающего актера.
Примечания:
Из-за ограничений синтаксиса название данной функции на официальном Wiki отображается не совсем корректно - в названии функции должен стоять символ подчеркивания (“_”), а не пробел.
Эта функция является эквивалентом консольной команды SetTargetRefractionFire (краткое название STRF) и будет работать так, как будто была вызвана из консоли.
Эта функция не была изначально рассчитана на использование в скриптах, поэтому может вести себя некорректно.
См. также: SetTargetRefraction
Относится к типу: OBSE Console Functions

Con_SexChange

Синтаксис:
CODE
actorRef.con_SexChange

Может быть вызвана через указатель actorRef . Изменяет пол вызывающего персонажа – мужчина становится женщиной и наоборот.

Примеры:
CODE
player.con_SexChange

В приведенном примере изменяется пол персонажа игрока.
Результаты тестирования в консоли:
Zomb: При вызове на плейере - работает. При вызове на НПС в ячейке с плеером игра вылетает.
Zig: Вылетало и на плейере и на нпс...
Примечания:
Из-за ограничений синтаксиса название данной функции на официальном Wiki отображается не совсем корректно - в названии функции должен стоять символ подчеркивания (“_”), а не пробел.
Эта функция является эквивалентом консольной команды SexChange и будет работать так, как будто была вызвана из консоли.
Эта функция не была изначально рассчитана на использование в скриптах, поэтому может вести себя некорректно.
Относится к типу: OBSE Console Functions | OBSE Reference Functions

Sin

Синтаксис:
CODE
sin [float]

Функция Sin возвращает синус угла (float).
Пример:
CODE
set y to r * (sin theta)

Примечания:
Начиная с версии OBSE v0005 все тригонометрические функции принимают и возвращают значения в градусах. Версии функций, работающих с радианами, все еще доступны, если к ним добавить префикс с буквой "R", т.е. RSin).
Эта функция будет работать в математической формуле (a + sin B), но если вы используете в качестве аргумента для этой функции математическую формулу, то эта функция компилироваться не будет (например, (sin (a + B)).
sin a + b = (sin a) + b
См. также: cos, tan, asin, acos, atan, atan2
Относится к типу: OBSE Math Functions | OBSE Trigonometric Functions

Sinh

Синтаксис:
CODE
sinh [float]

Функция Sinh возвращает гиперболический синус угла (float).
Пример:
CODE
set x to sinh theta

Примечания:
Начиная с версии OBSE v0005 все тригонометрические функции принимают и возвращают значения в градусах. Версии функций, работающих с радианами, все еще доступны, если к ним добавить префикс с буквой "R", т.е. RSinh).
Эта функция будет работать в математической формуле (a + sinh B), но если вы используете в качестве аргумента для этой функции математическую формулу, то эта функция компилироваться не будет (например, (sinh (a + B)).
sinh a + b = (sinh a) + b
См. также: sin, cos, tan, asin, acos, atan, cosh, tanh
Относится к типу: OBSE Math Functions | OBSE Trigonometric Functions

SquareRoot

Синтаксис:
CODE
SquareRoot [float]

CODE
sqrt [float]

Функция SquareRoot возвращает квадратный корень числа (float).
Пример:
CODE

set n to SquareRoot n

Относится к типу: OBSE Math Functions

T

Tan

Синтаксис:
CODE
tan [float]

Возвращает тангенс угла.
Пример:
CODE
set z to r * (tan theta)

(в отличие от функций sin и cos, пример для которых имел какой-то смысл, это пример – просто пример)
Примечания:
Начиная с v0005, все тригонометрические функции OBSE принимают и возвращают значения в градусах. Также доступны версии функций, которые принимают значения в радианах. Для их использования необходимо добавить префикс “r” (например, rtan)
Эта функция может быть использована в математической формуле (a + tan B), но математическая формулы в качестве аргумента для функции не подходит (tan (a + B) не работает).
tan a + b = (tan a) + b
См. также: sin, cos, asin, acos, atan, atan2
Относится к типу: OBSE Math Functions | OBSE Trigonometric Functions

Tanh

Синтаксис:
CODE
tanh [float]

Возвращает гиперболический тангенс угла.
CODE
set x to tanh theta

Примечания:
Начиная с v0005, все тригонометрические функции OBSE принимают и возвращают значения в градусах. Также доступны версии функций, которые принимают значения в радианах. Для их использования необходимо добавить префикс “r” (например, RTanh)
Эта функция может быть использована в математической формуле (a + tanh B), но математическая формулы в качестве аргумента для функции не подходит (tanh (a + B) не работает).
tanh a + b = (tanh a) + b
См. также: sin, cos, tan, asin, acos, atan, sinh, cosh
Относится к типу: OBSE Math Functions | OBSE Trigonometric Functions

TapKey

Синтаксис:
CODE
TapKey key

Функция TapKey сообщает игре, что игрок нажал на определенную клавишу клавиатуры (key), и игра соответствующим образом отреагирует. Функция использует сканкоды DirectX.
Пример:
CODE
TapKey 18; Нажатие на клавишу “E”

В приведенном примере персонаж игрока подпрыгнет (если игровое управление установлено по умолчанию)
Примечания:
Сканкоды DirectX обычно представляются в шестнадцатеричном формате, но эта функция принимает десятичные значения. Помимо обычных сканкодов DX, можно использовать числа от 256 до 263 для эмуляции нажатия на кнопки мыши, и от 264 до 265 - для движений колесика мыши.
Похоже, что функция TapKey не работает в режиме меню.

Перечень сканкодов DX приведен в приложении 3: Сканкоды DX

См. также: HoldKey, ReleaseKey, HammerKey, AHammerKey, UnhammerKey
Относится к типу: OBSE Input Functions

Con_ToggleDetection

Синтаксис:
CODE
con_ToggleDetection

То же самое, что и TDETECT.
Примечания:
Эта функция идентична консольной функции ToggleDetection и должна вести себя, как если бы вы вызвали функцию ToggleDetection в консоли.
Эта функция не предназначена для вызова из скриптов, поэтому может работать не так, как ожидается.
Относится к типу: OBSE Console Functions

U

UnhammerKey

Синтаксис:
CODE
UnhammerKey key

Отменяет действия вызванных ранее функций Turns HammerKey или AHammerKey. Эта функция использует сканкоды DirectX.
Пример:
CODE
if (timer == 0)
HammerKey 29;Left Control
elseif (timer < 30)
set timer to (timer + GetSecondsPassed)
else
UnhammerKey 29
endif

В приведенном примере персонаж игрока будет приседать и вставать (то есть красться и возвращаться в обычный режим) в течение 30 секунд.
Примечания:
Сканкоды DirectX обычно представляются в шестнадцатеричном формате, но эта функция принимает десятичные значения. Помимо обычных сканкодов DX, можно использовать числа от 256 до 263 для эмуляции нажатия на кнопки мыши, и от 264 до 265 для движений колесика мыши.
Перечень сканкодов DX приведен в приложении 3: Сканкоды DX
См. также: HammerKey, AHammerKey, TapKey, HoldKey, ReleaseKey
Относится к типу: OBSE Input Functions

W

Con_WaterDeepColor

Синтаксис:
CODE
con_WaterDeepColor red green blue

Функция con_WaterDeepColor изменяет цвет воды на мелководье с помощью параметров red, green и blue.
Примечания:
Эта функция идентична консольной функции WaterShallowColor, и должна вести себя так, как если бы вы набрали WaterShallowColor в консоли.
Эта функция не предназначена для вызова из скриптов, поэтому может работать не так, как ожидается.
Относится к типу: OBSE Console Functions

Con_WaterReflectionColor

Синтаксис:
CODE
con_WaterReflectionColor red green blue

Функция con_WaterReflectionColor изменяет цвет отражений в воде.

Примечания:
Эта функция идентична консольной функции WaterReflectionColor, и должна вести себя так, как если бы вы набрали WaterReflectionColor в консоли.
Эта функция не предназначена для вызова из скриптов, поэтому может работать не так, как ожидается.
Относится к типу: OBSE Console Functions

Con_WaterShallowColor

Синтаксис:
CODE
con_WaterShallowColor red green blue

Функция con_WaterShallowColor изменяет цвет воды в глубине с помощью параметров red, green и blue.
Примечания:
Эта функция идентична консольной функции WaterDeepColor, и должна вести себя так, как если бы вы набрали WaterDeepColor в консоли.
Эта функция не предназначена для вызова из скриптов, поэтому может работать не так, как ожидается.
Относится к типу: OBSE Console Functions

8. Console Functions

8.1 Общие сведения.

Эти функции будут доступны для использования при включенной игровой консоли. Вызывается консоль с помощью клавиши "~" (тильда).
Консольные функции (команды) не могут быть скомпилированы в скриптах в редакторе, если вы, конечно, не используете для этих целей OBSE. Правда, поддержка обеспечивается отнюдь не для всех консольных команд.

Вы можете "прокрутить" содержимое диалогового окна консоли, используя кнопки клавиатуры "PageUp" и "PageDown".
В категории Console Functions приведена ТАБЛИЦА, содержащая полный перечень консольных функций, но с весьма краткими описаниями.

Содержит 4 подкатегории, имеющие отношение к консоли:
Debug Text - Описание консольной команды "Toggle Debug Text" (или TDT). Следует особо отметить эту команду. Она является довольно объемной. Для нее даже выделена отдельная страница с подробным описанием. Пожалуй, она является главным отладочным средством.

HDR Settings
IsActorsAIOff
ToggleActorsAI

8.2 Полный перечень консольных функций.

В скобках приведено краткое обозначение.
Среди них нет ни одной совпадающей с обычными скриптовыми функциями.
Всего насчитывается 130 функций:
AddDecal (-)
AddFaceAnimNote (AFAN)
BeginTrace (bt)
BetaComment (BC)
CalcLowPathToPoint (LP2P)
CalcPathToPoint (P2P)
CenterOnCell (COC)
CenterOnExterior (COE)
CenterOnWorld (COW)
ClearAdaptedLight (cal)
CloseFile (-)
CompleteAllQuestStages (caqs)
DumpTexturePalette (DTP)
FlushNonPersistActors (Flush)
FreezeRenderAccumulation (fra)
GetINISetting (GetINI)
HairTint (-)
Help (-)
LoadGame (load)
ModWaterShader (mws)
MoveToQuestTarget (movetoqt)
OutputArchiveProfile (oap)
OutputLocalMapPictures (OLMP)

OutputMemContexts (omc)
OutputMemStats (oms)
PickRefByID (PRID)
PlayerSpellBook (psb)
PrintAiList (pai)
PrintHDRParam (PHP)
PrintNPCDialog (pdialog)
PurgeCellBuffers (pcb)
QuitGame (qqq)
RefreshINI (REFINI)
RefreshShaders (-)
ReloadCurrentClimate (rcc)
ReloadCurrentWeather (rcw)
ResetMemContexts (rmc)
RevertWorld (rw)
RunCellTest (rct)
RunMemoryPass (rmp)
SaveGame (save)
SaveIniFiles (saveini)
SaveWorld (-)
SetCameraFOV (FOV)
SetClipDist (-)
SetDebugText (sdt)
SetFog (-)
SetGameSetting (SetGS)
SetGamma (sg)
SetHDRParam (SHP)
SetImageSpaceGlow (SISG)
SetINISetting (SetINI)
SetLightingPasses (SLP)
SetSkyParam (SSP)
SetSTBBColorConstants (sscc)
SetTargetRefraction (str)
SetTargetRefractionFire (strf)
SetTreeMipmapBias (stmb)
SexChange (-)
Show (TST)
Show1stPerson (S1ST)
ShowAnim (SA)
ShowFullQuestLog (SFQL)
ShowHeadTrackTarget (SHeadT)
ShowPivot (SP)
ShowQuestLog (SQL)
ShowQuests (SQ)
ShowQuestTargets (SQT)
ShowQuestVars (SQV)
ShowRenderPasses (srp)
ShowScenegraph (SSG)
ShowSubSpaces (sss)
ShowSubtitle (-)
ShowVars (SV)
ShowWhoDetectsPlayer (SWDP)
SpeakSound (-)
StartAllQuests (saq)
TestAllCells (TAC)
TestCode (-)
TestLocalMap (tlm)
TestSeenData (tsd)
ToggleAI (TAI)
ToggleAiSchedules (TAIS)
ToggleBorders (TB)
ToggleCastShadows (tsh)
ToggleCellNode (TCN)
ToggleCharControllerShape (TCCS)
ToggleCollision (TCL)
ToggleCollisionGeometry (TCG)
ToggleCombatAI (tcai)
ToggleCombatStats (TCS)

ToggleConversations (TCONV)
ToggleDebugText (TDT)
ToggleDetection (TDETECT)
ToggleDetectionStats (tds)
ToggleEmotions (temo)
ToggleFlyCam (tfc)
ToggleFogOfWar (TFOW)
ToggleFullHelp (TFH)
ToggleGodMode (TGM)
ToggleGrass (TG)
ToggleGrassUpdate (TGU)
ToggleHDRDebug (THD)
ToggleHighProcess (THIGHPROCESS)
ToggleLeaves (TLV)
ToggleLiteBrite (tlb)
ToggleLODLand (TLL)
ToggleLowProcess (TLOWPROCESS)
ToggleMagicStats (TMS)
ToggleMapMarkers (tmm)
ToggleMaterialGeometry (TMG)
ToggleMenus (TM)
ToggleMiddleHighProcess (TMHIGHPROCESS)
ToggleMiddleLowProcess (TMLOWPROCESS)
ToggleOcclusion (tocc)
TogglePathGrid (TPG)
TogglePathLine (TPL)
ToggleRefractionDebug (trd)
ToggleSafeZone (TSZ)
ToggleScripts (TSCR)
ToggleShadowVolumes (TSV)
ToggleSky (TS)
ToggleTrees (TT)
ToggleWaterRadius (TWR)
ToggleWaterSystem (TWS)
ToggleWireframe (TWF)
Verbose (VERBOSE)
WaterDeepColor (deep)
WaterReflectionColor (refl)
WaterShallowColor (shallow)

Garin:
Примечание. С консольными функциями много еще неясного. Требуется тестирование. Я специально привел результаты тестирования, практически не изменяя текст. Может, это поможет вам разобраться с некоторыми непонятными функциями. Просьба ко всем, у кого есть полезная информация, кто работал с консолью и хорошо знает, о чем речь, сообщить нам. Мы с удовольствием включим в учебник и ваш опыт. Разумеется, с указанием вашего авторства.

8.3 Описания консольных функций (Console functions) в алфавитном порядке
Эти функции будут доступны для использования при включенной игровой консоли. Они не могут быть скомпилированы в скриптах в редакторе, разве что вы используете OBSE, да и то для ограниченного количества функций.
Вы можете "прокрутить" содержимое диалогового окна консоли, используя кнопки клавиатуры "PageUp" и "PageDown".

AddDecal
Краткое название: AddDecal
Вызывается на объекте: нет
Параметры: нет
Описание: нет
Zomb: Эффекта при использовании просто/на игроке/на плеере (в городе) не заметил.
Zig: Эффекта не обнаружил
Garin: Decal - бумага. Может, добавить бумагу или свитки/книги?

AddFaceAnimNote
Краткое название: AFAN
Вызывается на объекте: да
Параметры: sString
Описание: нет
Zomb: Эффекта при использовании просто/на игроке/на плеере (в городе) не заметил.
Zig: По идее это добавляет примечание для анимации лица выделеного персонажа, но вот как потом просмотреть этот коментраий я не обнаружил.

BeginTrace
Краткое название: bt
Вызывается на объекте: нет
Параметры: нет
Описание: Создает файл trace (только для xbox)

BetaComment
Краткое название: BC
Вызывается на объекте: нет
Параметры: sString
Описание: Добавляет комментарий к файлу, указанному в oblivion.ini в секции [General] 'sBetaCommentFile'.
Примечание: сначала выберите объект.
Пример: bc "This rock is too high."

CalcLowPathToPoint
Краткое название: LP2P
Вызывается на объекте: да
Параметры: bIgnoreLocks, bAllowDisabledDoors, bIgnoreMinUse
Описание: игнорировать замки (bIgnoreLocks), позволяет проходить через отключенные двери (bAllowDisabledDoors), игнорировать минимальное использование (bIgnoreMinUse).
GW: Правильно ли перевел - хз
Zomb: Работает на NPC. Выдаёт следующую инфу(частность):
- Travel 332 units in worldspace "SceengardWorld (000142E5)"
- Walk to coord (-65426 643 6448)
--Total distance:332 units.
--Estimated Travel Time: 0.01 game hour

CalcPathToPoint
Краткое название: P2P
Вызывается на объекте: да
Параметры: нет
Описание: нет
Zomb: Насколько я понял, рисует (а может генерирует) на земле цепочку вейпоинтов от места, где стоит плейер, до места, где стоит объект.
Zig: Небольшая поправка. Рисует путь Объекта до игрока. То есть как этот объект (NPC, Creature) будет двигаться к игроку.

CenterOnCell
Краткое название: COC
Вызывается на объекте: нет
Параметры: CellName
Описание: нет
Zomb: Перемещает плейера в центр указанной ячейки (функция аналогична TES3).

CenterOnExterior
Краткое название: COE
Вызывается на объекте: нет
Параметры: x, y
Описание: нет
Zomb: Перемещает плейера то ли в центр внешней ячейки с заданными координатами, то ли просто в место с заданными координатами.

CenterOnWorld
Краткое название: COW
Вызывается на объекте: нет
Параметры: WorldspaceName, CellCoordX, CellCoordY
Описание: Позиционирует игрока в мире WorldspaceName в ячейке с координатами CellCoordX, CellCoordY. Пример: COW worldname -10 5

ClearAdaptedLight
Краткое название: cal
Вызывается на объекте: нет
Параметры: нет
Описание: Clears the HDR adapted light texture
(GW - хз)
Zomb: Эффекта не заметил.
Zig: Сбрасывает HDR освещение на первоначальное состояние.

CloseFile
Краткое название: CloseFile
Вызывается на объекте: нет
Параметры: Filename
Описание: нет
Zomb: Бес знает, что в чём это выражается, но эта команада закрывает файлы. Срабатывает на плагинах, причём как загруженных, так и неподключенных. Файл credits.txt закрыть не удалось ^_^
Zig: Файл credits.txt закрыть удалось путем нажимания на красный кестик в левом верхнем углу окошечка :)

CompleteAllQuestStages
Краткое название: caqs
Вызывается на объекте: нет
Параметры: нет
Описание: Дает записи в журнал для всех стадий всех квестов.

DumpTexturePalette
Краткое название: DTP
Вызывается на объекте: нет
Параметры: SortType (optional)
Описание: Dump texture palette contents to warning file (param is sort type: f-filename,s-size,c-count)
Zomb: Эффекта не заметил. Никаких warring file'ов не нашёл.
Zig: Очевидно, команда была создана разработчиками для проверки "тупящих" текстур. Создаст варнинг файл, если будет обнаружена ошибка текстуры.

FlushNonPersistActors
Краткое название: Flush
Вызывается на объекте: нет
Параметры: iCount
Описание: Deletes all the actors in High who are not persistant
Zig: Удаляет всех актеров приоритетом ниже высокого, из обработки процессором.

FreezeRenderAccumulation
Краткое название: fra
Вызывается на объекте: нет
Параметры: нет
Описание: рендерит геометрию, видимую только в текущем фрейме.
Zomb: Показывает только те статики, которые находятся в кадре в момент вызова функции. Земля и всё остальное дисейблится. НПС продолжают ходить, как ни в чём не бывало. Окошко консоли не видно. Чтоб отключить эффект, надо "на ощупь" вызвать функцию ещё раз (~, вверх, enter).

GetINISetting
Краткое название: GetINI
Вызывается на объекте: нет
Параметры: SettingName
Zomb: Описание: Выводит в консоль названную строчку из Oblivion.ini

HairTint
Краткое название: HairTint
Вызывается на объекте: нет
Параметры: iRed, iGreen, iBlue
Описание: 3 целочисленных параметра, RGB
Zomb: По идее должен перекрашивать волосы, но ни на плейере, ни на НПС эффекта нет
Zig: Перекрашивает волосы, если в восьмизначной адресации цветов писать параметры.

Help
Краткое название: Help
Вызывается на объекте: нет
Параметры: нет
Описание: показывает эту справку.

LoadGame
Краткое название: load
Вызывается на объекте: нет
Параметры: SaveName
Описание: Загружает игру с именем <filename>
Zomb: Исправно загрузил мне autosave.
Zig: Подтверждаю. Работает только, если сэйв на английском. На русском в Обливионе набрать имя файла не получится :)

ModWaterShader
Краткое название: mws
Вызывается на объекте: нет
Параметры: String, Float (optional)
Описание: Изменяет параметры водных шейдеров

MoveToQuestTarget
Краткое название: movetoqt
Вызывается на объекте: нет
Параметры: QuestTargetNumber (не обязательно)
Описание: Перемещает игрока к текущей цели квеста (допол. параметр: число цели, от 1 до N).
Zomb: Как сказано выше. Прыжок в ячейку с зелёной стрелкой.

OutputArchiveProfile
Краткое название: oap
Вызывается на объекте: нет
Параметры: String (optional)
Описание: Выводит содержание архивного профайла в файл (Output Archive profile info to a file)
Zomb: Мне он ответил, что "Archive profile is not enabled." ("Активный профайл не включен")
Zig: Тоже самое.
"Активный профайл не включен"

OutputLocalMapPictures
Краткое название: OLMP
Вызывается на объекте: нет
Параметры: нет
Описание: Writes out the current local map.
Zomb: Эффекта не замечено, хотя над чем-то функция секунду думает. Локальная карта в инвенторе не обновляется (даже после выхода и входа в ячейку).
Zig: Функция должна была экспортировать изображения карт. Не работает.

OutputMemContexts
Краткое название: omc
Вызывается на объекте: нет
Параметры: Filename (optional)
Описание: Выводит содержимое памяти в файл (Output Mem Context info to a file)
Zomb: Мне он ответил, что "This function only work in MEM_DEBUG."
Zig: Мем дебаг, по идее, выискивает ошибки в оперативной памяти. Возможно. Но я ни в чем не уверен :)
Garin: "This function only work in MEM_DEBUG." - Примерный перевод: "Эта функция работает только в режиме отладки памяти."

OutputMemStats
Краткое название: oms
Вызывается на объекте: нет
Параметры: Filename (optional)
Описание: Выводит статистику памяти в файл (Output Mem Stats info to a file)
Zomb: "This function only work in MEM_DEBUG."
Garin: "This function only work in MEM_DEBUG." - Эта функция работает только в режиме отладки памяти.

PickRefByID
Краткое название: PRID
Вызывается на объекте: нет
Параметры: ObjectRefID
Описание: Выделяет копию для консоли по ее ID.
Zomb: Получилось вызвать только с параметром "player". Остальные ID не принимал. При вводе PRID Player в фокусе (сверху написано) появлялся сам плеер. И всё.
Zig: Работает, если указывать ссылочные id, а не общие. Это выглядит примерно так: prid 0001fc5c. Выделяет объект.

PlayerSpellBook
Краткое название: psb
Вызывается на объекте: нет
Параметры: нет
Описание: Добавляет игроку все заклинания.

PrintAiList
Краткое название: pai
Вызывается на объекте: нет
Параметры: нет

Описание: Печатает все списки AI.
Zomb: Пишет, что "AI list painted", но ни одного нового файла в папке с игрой или с профилем я не нашёл.
Zig: Может, мы не правильно поняли авторов? Возможно оно распечатывает на ПРИНТЕРЕ списки всех ai.

PrintHDRParam
Краткое название: PHP
Вызывается на объекте: нет
Параметры: нет
Описание: Печатает текущие настройки HDR.
Zomb: Выводит следующее сообщение:
Current HDR Params (текущие параметры HDR):
SISG: iNumBlurpasses=, fBlurRadius=, fBrightClamp=, fBrightScale=
SSP: fSunlightDimmer=, Lum Ramp=
SHP: fEyeAdaptSpeed=, fEmissiveHDRMult=, fTreeDimmer=, fGrassDimmer=, fUpperLUMClamp, fTargetLUM

PrintNPCDialog
Краткое название: pdialog
Вызывается на объекте: нет
Параметры: нет
Описание: Печатает диалог NPC
Zomb: Никакого эффекта ни в диалоге, ни при разговоре двух неписей.

PurgeCellBuffers
Краткое название: pcb
Вызывается на объекте: нет
Параметры: нет
Описание: Выгружает все несвязанные с игрой в данный момент ячейки из буфера. Другими словами, удаляет из памяти текстуры или модели, которые находятся вне загруженной в данный момент области. Использование этой функции очистит память и поможет в некоторых случаях увеличить быстродействие игры.

Данная возможность была добавлена первым официальным патчем (v1.1.511) в Oblivion.exe, но не в TESConstructionSet.exe, поэтому её можно вызвать из консоли, но редактор не признаёт её корректной и никогда не скомпилирует скрипт, содержащий данную функцию.

Oblivion Script Extender позволяет скриптерам использовать данную функцию, но, в отличие от остальных функций OBSE, её использование не требует наличия OBSE, так что плагины, использующие её, будут работать и у тех, кто OBSE не имеет .

В отличие от других функций OBSE, которые иммитируют консольные функции, эта функция была создана для использования в скриптах, и теперь это стало возможным.

QuitGame
Краткое название: qqq
Вызывается на объекте: нет
Параметры: нет
Описание: Немедленный выход их игры, не проходя все меню.

RefreshINI
Краткое название: REFINI
Вызывается на объекте: нет
Параметры: нет
Описание: Обновить установки INI из файла.

RefreshShaders
Краткое название: RefreshShaders
Вызывается на объекте: нет
Параметры: нет
Описание: Перезагрузить шейдеры HLSL с диска

ReloadCurrentClimate
Краткое название: rcc
Вызывается на объекте: нет
Параметры: нет
Описание: Перезагрузить значения из текущего климата

ReloadCurrentWeather
Краткое название: rcw
Вызывается на объекте: нет
Параметры: нет

Описание: Перезагрузить значения из текущей погоды

ResetMemContexts
Краткое название: rmc
Вызывается на объекте: нет
Параметры: нет
Описание: Reset Max Mem Contexts
Zomb: Выдает сообщение: "This function only work in MEM_DEBUG" (Эта функция работает только в режиме отладки памяти).

RevertWorld
Краткое название: rw
Вызывается на объекте: нет
Параметры: нет
Описание: Revert the world
Zomb: Начинается перезагрузка карты. После минут 20 мне это надоело... Тестил в Бруме и Скингарде.

RunCellTest
Краткое название: rct
Вызывается на объекте: нет
Параметры: iFlag (не обязательно)
Описание: Запускает тест ячейки
Zomb: Начинаются дикие прыжки по всем ячейкам игры.
Zig: Ячейки следуют в алфавитном порядке :)

RunMemoryPass
Краткое название: rmp
Вызывается на объекте: нет
Параметры: iFlag
Описание: Выполняется один проход очистки памяти.
Zomb: Эффекта не заметил

SaveGame
Краткое название: save
Вызывается на объекте: нет
Параметры: SaveName
Описание: Сохраняет игру в <filename>

SaveIniFiles
Краткое название: saveini
Вызывается на объекте: нет
Параметры: нет
Описание: Записывает все текущие параметры в файл *.ini

SaveWorld
Краткое название: SaveWorld
Вызывается на объекте: нет
Параметры: Filename
Описание: Сохраняет игровой мир в файл <filename>
Zomb: А вот это уже интересно!
После запуска в Бруме он мне создал в папке с игрой четырёхметровый файл с расширением *.hkx. В нём, в виде, кажется, xml (здесь могу наврать) выведено много структурированной информации по миру. Наверное, ВСЯ информация. Лично я пока в этом ничего не понял, но чувствую, что в этом файле есть большой потенциал ^_^
Zig: Абослютно простой файл, где содержится структура мира, перечисляется наличие использованых моделей, диалоги, спеллы и т.д. Работает и с плагинами. В Tes Construction Set подобное можно увидеть, если в меню загрузки плагинов выделить любой плагин и нажать "Details". Но там эта информация сильно упрощена.

SetCameraFOV
Краткое название: FOV
Вызывается на объекте: нет
Параметры: iDegrees
Описание: Меняет поле зрения камеры (в градусах): по умолчанию 75
Zomb: Охренительный спецэффект ^_^. Это должен заценить каждый.

SetClipDist
Краткое название: SetClipDist
Вызывается на объекте: нет
Параметры: fClipDistance

Описание: Команда SetClipDist с помощью параметра fClipDistance (тип float) устанавливает расстояние (в игровых единицах длины), на котором перестают определяться коллизии (Collision Detection). С использованием OBSE эта функция теперь может работать в скриптах (con_SetClipDist)
Garin: Краткая информация.
Напомним, что коллизии – это пересечение множества точек, принадлежащих к различным трехмерным телам в процессе их соприкосновения. Факт столкновения обычно проверяется, чтобы объекты вели себя корректно, с учетом физики, заложенной в игровой мир (Rigid Body Dynamics). Чем больше взаимодействующих объектов находится в заданном от камеры радиусе, тем ниже будет быстродействие. Точное определение пересечений требует полного пересчета всех треугольников моделей, что также довольно существенно снижает fps. Поэтому на практике используются различные оптимизированные алгоритмы.
Пример:
CODE
SetClipDist 4096

В приведенном примере дистанция установлена равной 4096 игровых единиц длины, что равно длине одной стороны квадрата внешней игровой ячейки.

SetDebugText
Краткое название: sdt
Вызывается на объекте: нет
Параметры: iDebugPage
Описание: Определяет, какой текст отладки будет показан.

SetFog
Краткое название: SetFog
Вызывается на объекте: нет
Параметры: fStartDepth, fEndDepth
Описание: принимает две переменные типа float, устанавливает начальную и конечную глубину тумана. Теперь может использоваться и как скриптовая функция, если у вас установлено OBSE.

SetGameSetting
Краткое название: SetGS
Вызывается на объекте: нет
Параметры: sGSName, sGSValue
Описание: нет
Zig: Устанавливает настройки игры.

SetGamma
Краткое название: sg
Вызывается на объекте: нет
Параметры: fValue
Описание: Устанавливает новую гамму (gamma ramp).
Zomb: Меняет яркость картинки. От "всё белым-бело" при 0,01, до "и пришла тьма" при 1000. Но пределов, кажется, нет. Стандартное значение у меня было где-то чуть меньше 1.

SetHDRParam
Краткое название: SHP
Вызывается на объекте: нет
Параметры: fEyeAdaptSpeed, fEmissiveHDRMult, fTreeDimmer, fGrassDimmer, fValue5, fValue6
Описание: Устанавливает различные значения для шейдера HDR

SetImageSpaceGlow
Краткое название: SISG
Вызывается на объекте: нет
Параметры: iValue1, fValue2, fValue3, fValue4
Описание: нет
Zig: Довольно ахренительный спецэффект. Наберите sisg 1 2 3 4 и Обливион станет заметно веселей

SetINISetting
Краткое название: SetINI
Вызывается на объекте: нет
Параметры: sININame, sINIValue
Описание: нет
Zig: Устанавливает настройки ini файла. Почему-то не работает.

SetLightingPasses
Краткое название: SLP
Вызывается на объекте: нет

Параметры: sBitFlag
Описание: diff|tex|spec ex: 1010
Zomb: Диапазон 0-6 невключительно, иначе вылет. Делает какие-то странные вещи с освещением статиков. ИМХО бесполезно.

SetSkyParam
Краткое название: SSP
Вызывается на объекте: нет
Параметры: fValue1, fValue2
Описание: Устанавливает различные значения для неба
Zomb: Мгновенного эффекта не заметил.

SetSTBBColorConstants
Краткое название: sscc
Вызывается на объекте: нет
Параметры: fValue1, fValue2, fValue3, fValue4
Описание: Show speedtree billboard color tweak constants.
Zomb: Мгновенного эффекта не заметил.

SetTargetRefraction
Краткое название: str
Вызывается на объекте: нет
Параметры: fValue
Описание: Устанавливает преломляющую способность цели
Zomb: Делает с целью то же самое, что было на моём скрине.

SetTargetRefractionFire
Краткое название: strf
Вызывается на объекте: нет
Параметры: fValue1, iValue2
Описание: Устанавливает преломляющее значение огня цели
Zomb: При наборе "strf 1 1" получил вылет игры. Вообще применит не получилось.
Zig: Никаких вылетов не наблюдал.

SetTreeMipmapBias
Краткое название: stmb
Вызывается на объекте: нет
Параметры: fValue1, fValue2
Описание: Set mipmap LOD bias values for tree billboards.
Zomb: На практике творит какие-то странные вещи с дистанционной отрисовкой.

SexChange
Краткое название: SexChange
Вызывается на объекте: да
Параметры: нет
Описание: Выбранный npc мужчина становится женщиной и наоборот.
Zomb: При вызове на плеере - работает. Пр вызове на НПС в ячейке с плеером - игра вылетает.
Zig: Вылетало и на плеере и на нпс...

Show
Краткое название: TST
Вызывается на объекте: нет
Параметры: VarName
Описание: Показывает значения скриптовых переменных: show gamedayspassed

Show1stPerson
Краткое название: S1ST
Вызывается на объекте: нет
Параметры: нет
Описание: Показывает модель от 1-го лица камерой от 3-го лица. Если игрок находится в режиме от 3-го лица, показывает оба.
Zomb: Показывает только руки и оружие.
Zig: Точнее, переключается на вид от первого лица

ShowAnim
Краткое название: SA
Вызывается на объекте: да
Параметры: нет

Описание: Показывает статус анимации и актера.
Zomb: Показывает следующие значения (частный случай):
---Actor Variables---
Process Level: High
Wants Weapon Drawn 0, Weapon Drawn 0
Life state: Alive
Sit/Sleep State: Норм
Knock State: Норм
Has RagDoll: Yes
Current Package Target: имя непися (код)
---Animation---
Anims Loading: 0
LowerBody-> //Idle

ShowFullQuestLog
Краткое название: SFQL
Вызывается на объекте: нет
Параметры: QuestID
Описание: Показывает все записи журнала для одного указанного квеста

ShowHeadTrackTarget
Краткое название: SHeadT
Вызывается на объекте: да
Параметры: нет
Описание: Показывает цель слежения головы npc if set from look function
Zomb: Функция мне отвечала, что цели слежения нет, несмотря на то, что слежение явно было.
Zig: То же самое. Слежение идет, функция молчит.

ShowPivot
Краткое название: SP
Вызывается на объекте: да
Параметры: нет
Zomb: Временно помещает желтый плюс на центре вращения объекта.

ShowQuestLog
Краткое название: SQL
Вызывается на объекте: нет
Параметры: bShowCompleted
Описание: Показывает журнал квеста. Флаг: 0=текущие квесты, 1=завершенные квесты

ShowQuests
Краткое название: SQ
Вызывается на объекте: нет
Параметры: нет
Описание: Выводит список всех квестов.

ShowQuestTargets
Краткое название: SQT
Вызывается на объекте: нет
Параметры: нет
Описание: Показывает текущие цели квеста
Zomb: Показывает имя текущего квеста в редакторе, номер стадии, код референской цели и код двери, за которой её искать.

ShowQuestVars
Краткое название: SQV
Вызывается на объекте: нет
Параметры: QuestID
Zomb: Показывает квестовые переменные, а также приоритет и статус квеста [sqv QuestID]

ShowRenderPasses
Краткое название: srp
Вызывается на объекте: нет
Параметры: нет
Описание: Рендер изображения для следующего фрейма.
(display render passes for the next frame)
Zomb: Открывает отдельное окно, в котором выводится иерархический список узлов сцены. На время работы окна Обливион замораживается. Окно открывается в фоне - переключаться Alt+Tab
Zig: Окно отображает все, что загружено в данном фрейме... модели и текстуры.

ShowScenegraph
Краткое название: SSG
Вызывается на объекте: нет
Параметры: нет
Описание: Создать окно с полноценной игровой графической сценой.
(Create a window with the full game scene graph).
Zomb: Аналогично ShowRenderPasses, но показывает ещё и информацию по миру и интерфейсу.

ShowSubSpaces
Краткое название: sss
Вызывается на объекте: нет
Параметры: нет
Описание: Временное отображение подпространства. (Temporarily displays subspaces).
Zomb: Эффекта не заметил.

ShowSubtitle
Краткое название: ShowSubtitle
Вызывается на объекте: нет
Параметры: bFlag
Описание: Показ всех диалоговых субтитров. (1 – показывать всегда, 0 – всегда прятать).

ShowVars
Краткое название: SV
Вызывается на объекте: да
Параметры: нет
Описание: Показать переменные на объекте.
Пример: player->sv

ShowWhoDetectsPlayer
Краткое название: SWDP
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Отображает на экране имена тех, кто видит игрока. Если игрок в режиме скрытности, то показывает только тех, кто его обнаруживает.

SpeakSound
Краткое название: SpeakSound
Вызывается на объекте: да
Параметры: Filename, iEmotion, iEmotionValue
Описание: нет.
Zig: Воспроизводит звуковой файл для выделенного объекта. Сам файл плюс анимация выражения лица НПС во время воспроизведения.

StartAllQuests
Краткое название: saq
Вызывается на объекте: нет
Параметры: нет
Описание: Старт всех квестов.

TestAllCells
Краткое название: TAC
Вызывается на объекте: нет
Параметры: iValue
Описание: Тестирование всех ячеек (0 - стоп, 1 - старт, 2 - интерьеры, 3 – текущий внешний экстерьер)

TestCode
Краткое название: TestCode
Вызывается на объекте: нет
Параметры: нет
Описание: нет
Zomb: Эффекта не заметил

TestLocalMap
Краткое название: tlm
Вызывается на объекте: нет
Параметры: iValue
Описание: Симуляция локальной карты (1 или 0 для FOW - включить или выключить)

Zomb: На некоторое время создаёт на полу под игроком текстуру с локальной картой. Выглядит это более ярко, чем в инвентаре. С флагом 0 показывает всю ячейку/мир, с флагом 1 показывает только открытую часть (Fog of war)
Zig: Бывают глюки в виде черного квадрата.

TestSeenData
Краткое название: tsd
Вызывается на объекте: нет
Параметры: нет
Описание: Визуальное отображение текущих данных для просмотра.
Zomb: На некоторое время генерирует на земле какую-то красную сетку.

ToggleAI
Краткое название: TAI
Вызывается на объекте: нет
Параметры: нет
Описание: нет
Garin: Выключает Al у неписей.
Zomb: Выключает AI у актеров. При этом они вообще ничего не делают, но если кого-то ударить в замороженном состоянии, то после разморозки он вам ответит. Функция может выключить как одного (выделенного) актера, так и вырубить вообще всех, если никто не выделен.

ToggleAiSchedules
Краткое название: TAIS
Вызывается на объекте: нет
Параметры: нет
Описание: нет
Zomb: Вырубает только часть "мозга" NPC - его расписание. Неписи идут куда шли, не меняя направления и долбясь головой в стену. Если напасть - ответят сразу же. Действует сразу на всё живое.

ToggleBorders
Краткое название: TB
Вызывается на объекте: нет
Параметры: нет
Описание: Отображает линии границ для каждой ячейки.

ToggleCastShadows
Краткое название: tsh
Вызывается на объекте: нет
Параметры: iShadowType [0-6]
Zomb: Описание: включает/выключает тени для разных групп объектов.

ToggleCellNode
Краткое название: TCN
Вызывается на объекте: нет
Параметры: iValue [0-5]
Описание: Переключатель 3D для дочерней вершины ячейки: 0 - актер, 1 - маркер, 2 - земля, 3 - вода, 4 - статика, 5 - активно.
Zomb: Убирает визуальное отображение соотв. объектов. Причём, если выбрать акторов, то их модели исчезают, а тени остаются. При этом NPC продолжают действовать.

ToggleCharControllerShape
Краткое название: TCCS
Вызывается на объекте: нет
Параметры:
Описание: Toggle char controller shape type.
Zomb: Эффекта не заметил.

ToggleCollision
Краткое название: TCL
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Это же левитация!!! А если серьёзно, эта команда отключает столкновения со всеми объектами - можно проходить сквозь стены, ну и летать, соответственно (земли-то тоже не чувствуешь).

ToggleCollisionGeometry
Краткое название: TCG
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Показывает геометрию столкновений. Проще говоря, показывает границы объектов.

ToggleCombatAI
Краткое название: tcai
Вызывается на объекте: нет
Параметры: нет
Описание: Вкл/откл ИИ боя

ToggleCombatStats
Краткое название: TCS
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Команда удалена. Используйте TDT и переключитесь на страницу боевой информации.

ToggleConversations
Краткое название: TCONV
Вызывается на объекте: нет
Параметры: нет
Описание: Вкл/откл параметры разговора.
Zomb: Говорит, что показывает/прячет статы диалогов. Только я их нигде не заметил.
Zig: Отключает беседы между персонажами. Вроде.

ToggleDebugText
Краткое название: TDT
Вызывается на объекте: нет
Параметры: нет
Описание: Показывает fps и всевозможные числовые значения для отладки, отображая их на экране в консоли.

Debug Text (отладочный текст)
Перевод: Gwathlobal.

Консольная команда "Toggle Debug Text" (или TDT) может быть неоценимым инструментом для моддеров и игроков. Она может предоставить информацию, которую нельзя получить никакими другими средствами.
Чтобы выбрать, какая информация должна предоставляться в режиме отладки, существует несколько способов:
Зайдите в папку \MyDocuments\MyGames\Oblivion\ и откройте файл "Oblivion.ini". В нем найдите строку "iDebugText=", после которой следует число. Изменяя это число, вы можете изменять выводимую информацию.
Более простой способ – в игре в консоли напечатайте "sdt" плюс номер, который вы хотите использовать;
Третий способ - можно прокручивать страницы с отладочным текстом, используя клавишу Scroll Lock.
Вне зависимости от страницы, отладочный текст включает в себя счетчик FPS.
Числа, имеющие наибольшее применение для игроков и моддеров, это 0 и 10. Установив iDebugText в 10, вы узнаете, сколько повышений навыков имел игрок после последнего повышения уровня и каков будет множитель для характеристик в данный момент. Установив iDebugText в 0, модмейкеры могут узнать имя и ID ячейки, в которой находятся.

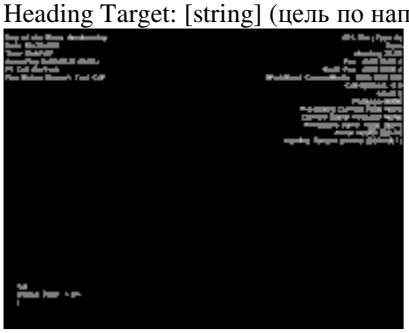
Вот перечень значений, которые могут использоваться в файле *.ini. Иногда выводимый текст меняется в зависимости от обстоятельств. Такие случаи будут отмечены особо.

Отрицательные числа
Любое отрицательное число приведет к выводу следующего текста:
Exterior Cell Buffer: [int] (буфер экстерьеров)
Interior Cell Buffer: [int] (буфер интерьеров)
High Actors: [int] (актеры с высоким приоритетом (?))
Middle High Actors: [int] (актеры со средне-высоким приоритетом (?))
Combat Actors: [int] (актеры в бою)
Non-Actor Mobile Objects: [int] (мобильные объекты не-актеры)

[Примечание: То же самое, что и sdt 32 (Oblivion v1.1 beta patch)]

0 – Время и место
Установка iDebugText в 0 приведет к выводу следующего текста:
[ЛЕВАЯ СТОРОНА]

Day of the Week: [in-game day] (день недели)
Date: [in-game date] (число календаря)
Time: [in-game time] (время)
Gameplay [time spent since last load] (время после последней загрузки)
PC Cell: [ID of current cell] (ID текущей ячейки)
Map Name: [Name of current cell] (Имя текущей ячейки)
[ПРАВАЯ СТОРОНА]
[player name] (имя игрока)
Heading: [float] (направление взгляда (?))
Pos: [x,y,z] (координаты)
Bip01 Pos: [x,y,z] (координаты Bip01)
WorldRoot CameraNode: [x,y,z] (координаты камеры)
Cell [cell ID]: [x,y] (ID ячейки и ее координаты)
Level: [int] (уровень)
Package: [string] (пакет)
Procedure Current Pack: [string] (процедура текущей упаковки)
Current Editor Package: [string] (текущий пакет редактира)
Procedure Editor Pack: [string] (процедура упаковки редактора)
Actor Health: [double] (здоровье актера)
Heading Target: [string] (цель по направлению взгляда)



Примечание: То же самое, что и sdt 33 (Oblivion v1.1 beta patch)

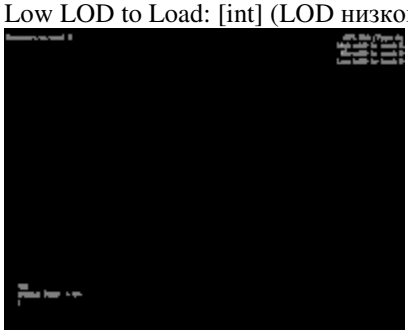
1 - Анимации
Установка iDebugText в 1 приведет к выводу следующего текста:
[actor name] (имя актера)
HK_STATE-> [string] (???)
BoneLOD [int] of [int] (???)
1st Person (в режиме от первого лица)
Lower Body -> //[Anim Type], Count: [int] (информация о нижней части тела)
time [float] move [float] attack [float] speed [int]/[int] delta [float]
[path to Animation] time [float]/[float] state [string] ease [float]
3rd Person (в режиме от третьего лица)
Lower Body -> //[Anim Type], Count: [int] (информация о нижней части тела)
time [float] move [float] attack [float] speed [int]/[int] delta [float]
[path to Animation] time [float]/[float] state [string] ease [float]

2 - установка в 2
Установка iDebugText в 2 приведет к выводу следующего текста:
Queued Count: [int]
Sleep Milliseconds: [int] (бездействие в милисекундах)
Count Mult: [float]
FPS Mult [float] (множитель FPS)

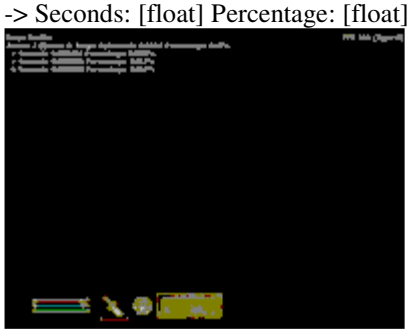
3 - установка в 3
Установка iDebugText в 3 приведет к выводу того же текста, что и 2.

4 – LOD экстерьеров
Установка iDebugText в 4 приведет к выводу следующего текста:

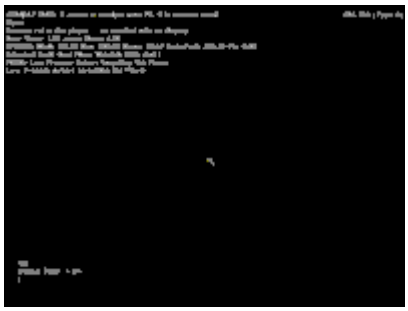
Exteriors to load: [int] (количество экстерьеров, которые необходимо загрузить)
[Exteriors being currently loaded] (экстерьеры, загруженные сейчас)
High LOD to Load: [int] (LOD высокой детализации, которые необходимо загрузить)
Mid LOD to Load: [int] (LOD средней детализации, которые необходимо загрузить)
Low LOD to Load: [int] (LOD низкой детализации, которые необходимо загрузить)



5 - Скрипты
Установка iDebugText в 5 приведет к выводу следующего текста:
Script Profiler (Профайлер скриптов)
Active: [int] (Quest: [int], Magic: [int]) Seconds: [float] Percentage: [float]% (Активные: (Квесты, Магия) Секунды, Процент)
{для каждого активного скрипта}
-> Seconds: [float] Percentage: [float]% (секунды, процент)



6 – Боевая информация
Установка iDebugText в 6 приведет к выводу следующего текста, когда в консольном режиме выделен игрок, или ничего не выделено:
COMBAT INFO: [int] actors in combat with PC, [int] in combat total (кол-во актеров, сражающихся с игроком, всего сражающихся актеров)
[player name] (имя игрока)
Current ref is the player - no combat info to display (текущий ref – игрок, нет информации)
Bow Timer: [float] Zoom timer: [float] (Таймер лука, таймер приближения)
SPEEDS: Walk: [float] Run: [float] Swim: [float] Swimfast: [float] Fly: [float] (скорость ходьбы, бега, плавания, быстрого плавания, полета)
High Process Actors targeting Player (Актеры высокого приоритета, нацеливающиеся на игрока)
[list of actors] (список актеров)
Middle Low Process Actors targeting Player (Актеры соедне-низкого приоритета, нацеливающиеся на игрока)
[list of actors] (список актеров)
Low Process Actors targeting Player (Актеры низкого приоритета, нацеливающиеся на игрока)
[list of actors] (список актеров)



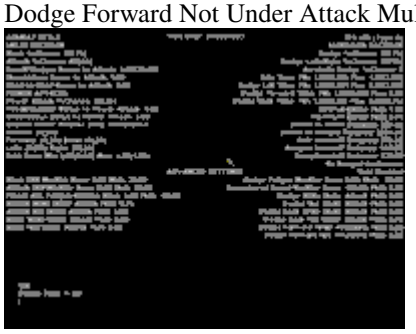
Установка iDebugText в 6 приведет к выводу следующего текста, когда в консольном режиме выделен враг:[list]
COMBAT INFO: [int] actors in combat with PC, [int] in combat total (кол-во актеров, сражающихся с игроком, всего сражающихся актеров)
[actor name] (имя актера)
H:[float]/[int] F:[float]/[int] M:[float]/[int] (Здоровье (тек/макс), Усталость (тек/макс), Магия (тек/макс))
Disposition to Opponent: [int] (отношение к оппоненту)
Distance: [float] Position: [float]deg (расстояние, положение в градусах)
Attack Reach: [float] (дистанция атаки)

Strategy: [string] (стратегия)
Maneuver: [string] [float]/[float] pkg radius/tgt val: [int]/[int] (меневр)
Melee: [string] (ближний бой)
Selected Spells (выбранные заклинания)[list]
[list] (список)
• Available Spells (доступные заклинания)
[list] (список)
• FORWARD Power Attack Range: [float] (дальность силовой атаки вперед)
[Can/Cannot] see main target (Может/не может видеть главную цель)
[segment] in View (сегмент в поле зрения)
Detection level on current target: [signed int] {if <0 then->}(UNDETECTED) (уровень заметности цели, если 0 - незаметна)
Current Targets (текущие цели)
[target name]: [int] (имя цели)
Bounds:(x,y,z)-(x,y,z)
ALLIES (союзники)
[list] (список)
[player name] (имя игрока)
H:[float]/[int] F:[float]/[int] M:[float]/[int] (Здоровье (тек/макс), Усталость (тек/макс), Магия (тек/макс))
Disposition to Opponent: [int] (отношение к оппоненту)
Target is [name]. {if PLAYER->}No AI Info (имя цели, если игрок – нет информации)

7 – Стиль боя
Установка iDebugText в 7 приведет к выводу следующего текста, когда в консольном режиме выделен игрок, или ничего не выделено:
COMBAT STYLE: Current Ref is the Player (текущий объект - игрок)

Установка iDebugText в 7 приведет к выводу следующего текста, когда в консольном режиме выделен враг:
COMBAT STYLE:
MELEE DECISION
Block %Chance: [int]([int]) (шанс блокирования)
Attack %Chance: [int]([int]) (шанс атаки)
Recoil/Stagger Bonus to Attack: [float]([int]) (бонус отскока/шатания к атаке)
Unconscious Bonus to Attack: [float]
Hand-To-Hand Bonus to Attack: [float] (рукопашный бонус к атаке)
POWER ATTACKS
Power Attack %Chance: [int]([int]) (шанс силовой атаки)
Recoil/Stagger Bonus to Power Attack: [float] (бонус отскока/шатания к силовой атаке)
Unconscious Bonus to Power Attack: [float]
Choose Power Attacks using %%Chance. (выбор силовой атаки, используя шанс %)
Normal: [int]([int]) (нормальная силовая атака)
Forward: [int]([int]) Back: [int]([int]) (силовая атака вперед, назад)
Left: [int]([int]) Right: [int]([int]) (силовая атака влево, вправо)
Hold Timer Min: [float]([float]) Max: [float]([float])
MANEUVER DECISION:
Dodge %Chance: [int]([int]) (шанс на уклонение)
Dodge Left/Right %Chance: [int]([int]) (шанс на уклонение влево/вправо)
Acrobatic Dodge %Chance: [int] (шанс на акробатическое уклонение)
Idle Timer Min: [float]([float]) Max: [float]([float])
Dodge L/R Timer Min: [float]([float]) Max: [float]([float])

Dodge Forward Timer Min: [float]([float]) Max: [float]
Dodge Back Timer Min: [float]([float]) Max: [float]([float])
Optimal Range Mult: [float] (множитель оптимальной дистанции)
Maximum Range Mult: [float] (множитель максимальной дистанции)
Switch to Melee Distance: [float] (переход к дистанции ближнего боя)
Switch to Ranged Distance: [float] (переход к дистанции дальнего боя)
Buff Standoff Distance: [float]
Ranged Standoff Distance: [float]
GroupStandoff Distance: [float]
[ranged preference string] (строка предпочтения дальнего боя)
ADVANCED DESICION
Yield [enabled/disabled] (поддавки вкл/выкл)
Block Skill Modifier Base: [float] Mult: [float] (базовый модификатор, множитель умения блокировать)
Attack Skill Modifier Base: [float] Mult: [float] (базовый модификатор, множитель умения атаковать)
Power Att. Fatigue Modifier Base: [float] Mult: [float] (базовый модификатор, множитель к усталости, затрачиваемой на силовую атаку)
Attack While Under Attack Mult: [float] (множитель к атаке, когда нападает враг)
Attack Not Under Attack Mult: [float] (множитель к атаке, когда не нападает враг)
Block While Under Attack Mult: [float] (множитель к блоку, когда нападает враг)
Block Not Under Attack Mult: [float] (множитель к блоку, когда не нападает враг)
Dodge Fatigue Modifier Base: [float] Mult: [float] (базовый модификатор, множитель умения уклоняться)
Encumbered Speed Modifier Base: [float] Mult: [float] (базовый модификатор, множитель скорости при нагрузке)
Dodge While Under Attack Mult: [float] (множитель к уклонению, когда нападает враг)
Dodge Not Under Attack Mult: [float] (множитель к уклонению, когда не нападает враг)
Dodge Back While Under Attack Mult: [float] (множитель к уклонению назад, когда нападает враг)
Dodge Back Not Under Attack Mult: [float] (множитель к уклонению назад, когда не нападает враг)
Dodge Forward While Under Attack Mult: [float] (множитель к уклонению вперед, когда нападает враг)
Dodge Forward Not Under Attack Mult: [float] (множитель к уклонению вперед, когда не нападает враг)



8 – Информация о магии
Установка iDebugText в 8 приведет к выводу следующего текста:
MAGIC INFO
[player name] (имя игрока)
Caster Magicka: [int]/[int] (магия заклинателя)
Caster [Inactive/Active] (заклинатель активен/неактивен)
CURRENT EFFECTS
[effect name]: Mag=[float] [Stat]=[int]/[int] from [location] (текущий эффект: магия, хар-ка, локация)

9 – Информация о актере
Установка iDebugText в 9 приведет к выводу следующего текста:
ACTOR INFO: [actor name] (имя актера)
ATTRIBUTES
Strength: [float]/[int] (сила)
Intelligence: [float]/[int] (интеллект)
Willpower: [float]/[int] (сила воли)
Agility: [float]/[int] (ловкость)
Speed: [float]/[int] (скорость)

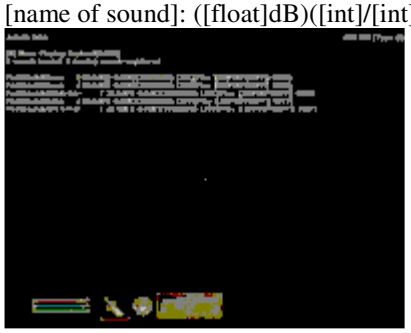
Endurance: [float]/[int] (выносливость)
Personality: [float]/[int] (обаяние)
Luck: [float]/[int] (удача)
Health: [float]/[int] (здоровье)
Magicka: [float]/[int] (магия)
Fatigue: [float]/[int] (усталость)
Encumbrance: [float]/[int] (нагрузка)
Aggression: [float]/[int] (агрессия)
Confidence: [float]/[int] (уверенность)
Energy: [float]/[int] (энергия)
Responsibility: [float]/[int] (ответственность)
Bounty: [float]/[int] (награда)
Fame: [float]/[int] (слава)
Infamy: [float]/[int] (дурная слава)
SKILLS
Armorer: [float]/[int] (оружейник)
Athletics: [float]/[int] (атлетика)
Blade: [float]/[int] (мечи)
Block: [float]/[int] (блок)
Blunt: [float]/[int] (дробящее оружие)
Hand to Hand: [float]/[int] (рукопашная)
Heavy Armor: [float]/[int] (тяжелая броня)
Alchemy: [float]/[int] (алхимия)
Alteration: [float]/[int] (изменения)
Conjuration: [float]/[int] (призывание)
Destruction: [float]/[int] (разрушение)
Illusion: [float]/[int] (иллюзии)
Mysticism: [float]/[int] (мистицизм)
Restoration: [float]/[int] (восстановление)
Acrobatics: [float]/[int] (акробатика)
Light Armor: [float]/[int] (легкая броня)
Marksman: [float]/[int] (стрельба)
Mercantile: [float]/[int] (торговля)
Security: [float]/[int] (взлом)
Sneak: [float]/[int] (скрытность)
Speechcraft: [float]/[int] (красноречие)
INVENTORY
[equipped item name]: [damage if weapon] [float]% (имя экипированного предмета, если оружие –повреждение)
ACTOR VALUES
Attack Bonus: [float]/[int] (бонус к атаке)
Blindness: [float]/[int] (слепота)
Paralysis: [float]/[int] (паралич)
Detect Item Range: [float]/[int] (дальность обнаружения объектов)
Swim Speed Multiplier: [float]/[int] (множитель скорости плавания)
Stunted Magicka: [float]/[int] (низкая магия)
Telekinesis: [float]/[int] (телекинез)
Resist Disease: [float]/[int] (сопротивление болезням)
Resist Paralysis: [float]/[int] (сопротивление параличу)
Vampirism: [float]/[int] (вампиризм)
Magicka Multiplier: [float]/[int] (множитель магии)
Defend Bonus: [float]/[int] (бонус к защите)
Chameleon: [float]/[int] (хамелеон)
Silence: [float]/[int] (тишина)
Spell Absorb Chance: [float]/[int] (шанс на поглощение заклинаний)
Water Breathing: [float]/[int] (дыхание под водой)
Detect Life Range: [float]/[int] (дальность обнаружения жизни)
Resist Fire: [float]/[int] (сопротивление огню)
Resist Magic: [float]/[int] (сопротивление магии)
Resist Poision: [float]/[int] (сопротивление яду)
Darkness: [float]/[int] (тьма)
Night Eye Bonus: [float]/[int] (бонус ночного зрения)
Casting Penalty: [float]/[int] (неуспех колдовства)
Invisibility: [float]/[int] (невидимость)
Confusion: [float]/[int] (конфузия)
Spell Reflect Chance: [float]/[int] (шанс на отражение заклинаний)
Water Walking: [float]/[int] (хождение по воде)
Reflect Damage: [float]/[int] (отражение повреждений)
Resist Frost: [float]/[int] (сопротивление холоду)

Resist Normal Weapons: [float]/[int] (сопротивление обычному оружию)
Resist Shock: [float]/[int] (сопротивление электричеству)
Resist Water Damage: [float]/[int] (сопротивление водным повреждениям)

10 – Повышение умений за уровень
Установка iDebugText в 10 приведет к выводу следующего текста:
PLAYER CHARACTER
Skill Usage
Armorer ([int]): advances: [int], usage [float]/[float] (оружейник)
Athletics ([int]): advances: [int], usage [float]/[float] (атлетика)
Blade ([int]): advances: [int], usage [float]/[float] (мечи)
Block ([int]): advances: [int], usage [float]/[float] (парирование)
Blunt ([int]): advances: [int], usage [float]/[float] (дробящее оружие)
Hand To Hand ([int]): advances: [int], usage [float]/[float] (рукопашная)
Heavy Armor ([int]): advances: [int], usage [float]/[float] (тяжелая броня)
Alchemy ([int]): advances: [int], usage [float]/[float] (алхимия)
Alteration ([int]): advances: [int], usage [float]/[float] (изменения)
Conjuration ([int]): advances: [int], usage [float]/[float] (призывание)
Destruction ([int]): advances: [int], usage [float]/[float] (разрушение)
Illusion ([int]): advances: [int], usage [float]/[float] (иллюзия)
Mysticism ([int]): advances: [int], usage [float]/[float] (мистицизм)
Restoration ([int]): advances: [int], usage [float]/[float] (восстановление)
Acrobatics ([int]): advances: [int], usage [float]/[float] (акробатика)
Light Armor ([int]): advances: [int], usage [float]/[float] (легкая броня)
Marksman ([int]): advances: [int], usage [float]/[float] (стрельба)
Mercantile ([int]): advances: [int], usage [float]/[float] (торговля)
Security ([int]): advances: [int], usage [float]/[float] (взлом)
Sneak ([int]): advances: [int], usage [float]/[float] (скрытность)
Speechcraft ([int]): advances: [int], usage [float]/[float] (красноречие)
Major Skills Advanced: [int]/[int] (повышение главных навыков)
Attribute Skill Counts
Advancement #[int]
Strength: [int] (повышение силы)
Intelligence: [int] (повышение интеллекта)
Willpower: [int] (повышение силы воли)
Agility: [int] (повышение ловкости)
Speed: [int] (повышение скорости)
Endurance: [int] (повышение выносливости)
Personality: [int] (повышение обаяния)
Luck: [int] (повышение удачи)
Specialization Counts
Combat: [int] (счетчик боевой специализации)
Magic: [int] (счетчик магической специализации)
Stealth: [int] (счетчик специализации в скрытности)

11 – Аудио информация
Установка iDebugText в 11 приведет к выводу следующего текста:
AUDIO INFO

[char] Music Playing: [string]([float]) (текущая музыка)
[int] sounds loaded. [int] moving sounds registered. (кол-во загруженных звуков, кол-во зарегистрированных движущихся звуков)
[name of sound]: ([float]dB)([int]/[int]) ([int]) [additional info] (имя звука: громкость)



12 - FPS
Установка iDebugText в 12 приведет к выводу следующего текста:
FPS: [int]

13 - Геометрия
Установка iDebugText в 13 приведет к выводу следующего текста:
Geometry [int] ([int])
Tri [int] : [float]
Pass [int] : [float]
TriPasses [int]
QueueMem [float] kb
TextureMem S [float] + R [float] = T [float] Mb
Occlusion Geom: [int], [int] tri, [int] wait loops
Sun Occlusion Wait Frames: [int]
Sun Occlusion Pixels: [int]
Bound Volume Occlusion Wait Loops: [int]
Active Lights: [int]/[int]
Grass: [int] g, [int] i
DistantLOD: [int] g, [int] i

14 и выше
Установка iDebugText в 14 и выше не приведет к выводу полезной информации или выведет информацию, как iDebugText 0:
Протестировано на Oblivion v1.1 beta patch:
sdt 14-23 включительно показывает 10 страниц данных об исходных текстурах. На тестовом компьютере данные сменялись раз в секунду, и их трудно было прочитать с экрана.
sdt 24 показывает надпись "Profiler Not Enabled" и FPS.
sdt 25 показывает надпись "Profiler(MAX) Not Enabled" и FPS.
sdt 26 показывает надпись "Heap Stats" и FPS.
sdt 27 показывает надпись "Mem Context Not Enabled" и FPS.
sdt 28 показывает надпись "SystemMem Context Not Enabled" и FPS.
sdt 29 показывает "SaveGame Info", относящиеся к сохранениям и FPS.
sdt 30 показывает "Paths to Build", FPS и данные "Paths Completed".
sdt 31 показывает информацию о локации и FPS.

sdt 32 показывает то же, что и sdt с отрицательными значениями.
sdt 33 показывает то же, что и sdt 0.
sdt больше 33 показывает то же, что и sdt 33. (протестировано до "sdt 40".)
[Примечание: Требуется подтверждение информации для Oblivion v1.1 beta patch. Требуется подтверждение, могут ли определенные страницы отладочной информации быть открыты с помощью oblivion.ini]

ToggleDetection
Краткое название: TDETECT
Вызывается на объекте: нет
Параметры: нет
Описание: нет
Zomb: Включает/выключает AI DETECTION. Неписи перестают друг друга видеть и не вступают в разговоры между собой.

ToggleDetectionStats
Краткое название: tds
Вызывается на объекте: нет
Параметры: нет
Описание: Показывает характеристики обнаружения у текущей копии.
Zomb: Эффекта не заметил

ToggleEmotions
Краткое название: temo
Вызывается на объекте: нет
Параметры: нет
Описание: Вкл/откл эмоции на лице NPC.

ToggleFlyCam
Краткое название: tfc
Вызывается на объекте: нет
Параметры: нет
Описание: Вкл/откл свободно летающую камеру (камеру НЛО).
Zomb: Тоже классная штука! Плейер стоит на месте, а вы управляете движением камеры. Можно полетать, посмотреть, что где творится. Камера проходит сквозь стены и землю.
Zig: Называется свободная камера. Незаменимая вешь для "мувикоделов".

ToggleFogOfWar
Краткое название: TFOW
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Вкл/откл туман войны на локальной карте.

ToggleFullHelp
Краткое название: TFH
Вызывается на объекте: нет
Параметры: нет
Описание: Вкл/откл полную справку
Zomb: Если кликнуть по объекту конольной стрелкой, показывает номер скрипта, владельца, какие-то флаги и др. информацию про объект.

ToggleGodMode
Краткое название: TGM
Вызывается на объекте: нет
Параметры: нет
Описание: Вкл/откл режим бога

ToggleGrass
Краткое название: TG
Вызывается на объекте: нет
Параметры: нет
Описание: Вкл/откл отображение травы.

ToggleGrassUpdate
Краткое название: TGU
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: отключает дорисовку травы вокруг игрока. Срабатывает в новой ячейке.

ToggleHDRDebug

Краткое название: THD
Вызывается на объекте: нет
Параметры: нет
Описание: Вкл/откл отладочные текстуры HDR.
Zomb: Показывает в углу картинку экрана с выделенными участками, обрабатываемыми HDR (наверно)
Zig: Подтверждаю.

ToggleHighProcess
Краткое название: THIGHPROCESS
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: включает/выключает IA для неписей с Hi Processing

ToggleLeaves
Краткое название: TLV
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Включает/выключает листву на деревьях и кустах.

ToggleLiteBrite
Краткое название: tlb
Вызывается на объекте: нет
Параметры: нет
Описание: Toggles lite brite render mode.
Zomb: Создаёт на земле какую-то серо-белую поверхность. Она неоднородная по тону и не совсем плоская. Что обозначает, я не понял. Если бы это был Морровинд, то я подумал бы, что она показывает вертексы тени.

ToggleLODLand
Краткое название: TLL
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Включает/выключает работу LOD. При выключенном LOD отключается отрисовка дальних земель и опускается туман.

ToggleLowProcess
Краткое название: TLOWPROCESS
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Аналогично ToggleHighProcess, но отключает ИИ для LOW PROCESS.

ToggleMagicStats
Краткое название: TMS
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Включает/выключает статистики по магии (я не заметил работы)

ToggleMapMarkers
Краткое название: tmm
Вызывается на объекте: нет
Параметры: iValue
Zomb: Описание: Вкл/откл маркеры локаций на глобальной карте (1-показывает все, 0-скрывает все).

ToggleMaterialGeometry
Краткое название: TMG
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Показывает материальную геометрию. Очень похоже на TCG.

ToggleMenus
Краткое название: TM
Вызывается на объекте: нет
Параметры: нет
Описание: Скрывает все меню. Используется для создания скриншотов.

ToggleMiddleHighProcess
Краткое название: TMHIGHPROCESS
Вызывается на объекте: нет
Параметры: нет

Zomb: Описание: Аналогично ToggleHighProcess, но отключает ИИ для MiddleHigh Process.

ToggleMiddleLowProcess
Краткое название: TMLOWPROCESS
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Аналогично ToggleHighProcess, но отключает ИИ для MiddleLow Process.

ToggleOcclusion
Краткое название: tocc
Вызывается на объекте: нет
Параметры: нет
Описание: toggle occlusion query for geometry
Zomb: Не совсем понял смысла. Под воздействием света (у меня был факел) начинают пропадать части моделей актеров.
Zig: Команда почему сделала Обливион похожей на Морроувинд.

TogglePathGrid
Краткое название: TPG
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Включает/выключает показ сети путевых точек (PathGrid)

TogglePathLine
Краткое название: TPL
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: На некоторое время включает отображение пути актора из ближайших путевых точек и цели движения.

ToggleRefractionDebug
Краткое название: trd
Вызывается на объекте: нет
Параметры: нет
Описание: Toggles refraction debug render texture.
Zomb: Эффекта не заметил

ToggleSafeZone
Краткое название: TSZ
Вызывается на объекте: нет
Параметры: нет
Описание: Display the television 85% safe zone.
Zomb: Эффекта не заметил

ToggleScripts
Краткое название: TSCR
Вызывается на объекте: нет
Параметры: нет
Описание: Вкл/откл исполнение скриптов.

ToggleShadowVolumes
Краткое название: TSV
Вызывается на объекте: нет
Параметры: нет
Описание: нет
Zomb: Эффекта не заметил

ToggleSky
Краткое название: TS
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Вкл/выкл отрисовку неба

ToggleTrees
Краткое название: TT
Вызывается на объекте: нет
Параметры: нет
Описание: Вкл/откл деревья

ToggleWaterRadius

Краткое название: TWR
Вызывается на объекте: нет
Параметры: нет
Описание: нет
Zomb: Показывает какие-то маркеры на поверхности земли и воды. Если вода близко - показывает на неё направление.

ToggleWaterSystem
Краткое название: TWS
Вызывается на объекте: нет
Параметры: нет
Zomb: Описание: Вкл/откл графическую систему воды.

ToggleWireframe
Краткое название: TWF
Вызывается на объекте: нет
Параметры: нет
Описание: Отображает мир в виде проволочного каркаса.

Verbose
Краткое название: VERBOSE
Вызывается на объекте: нет
Параметры: нет
Описание: Вкл/откл многословные сообщения ИИ/боя

WaterDeepColor
Краткое название: deep
Вызывается на объекте: нет
Параметры: iValue1, iValue2, iValue3
Описание: Изменяет глубину цвета воды

WaterReflectionColor
Краткое название: refl
Вызывается на объекте: нет
Параметры: iValue1, iValue2, iValue3
Описание: Изменяет цвета преломления воды

WaterShallowColor
Краткое название: shallow
Вызывается на объекте: нет
Параметры: iValue1, iValue2, iValue3
Описание: Изменяет цвет воды на мелководье

9.1 Приложение 1: Возвращаемые типы величин (OBSEValueTypes)

CODE
Диапазон - тип величины
Range - Value Types

000-099 - Общие значения (Common Values)
100-149 - Оружие и стрелы (Weapon and Ammo)
150-199 - Броня (Armor)
200-209 - Камни душ (Soul Gem)
210-219 - Ингредиенты (Ingredient)
220-229 - Алхимические предметы (Alchemy Item)

CODE

Num	GetOV	GetCV	Value
0	**f**		**weight - вес**
1	**i**		**gold value - количество золота (не считая зачарованных значений)**
2	**i**	**f**	**health (здоровье)**
3	**i**		**equipment SlotID (слот экипировки SlotID)**
4	**i**	**f**	**enchantment charge - (стоимость зачарования)**
5	**bis**		**quest item (квестовый предмет)**
6	**ref**		**enchantment (зачарование)**
100	**i**		**attack damage (ущерб от атаки)**
101	**f**		**reach (достижения)**

102 f speed (скорость)
103 i weapon type (тип оружия)
104 b ignores weapon resistence (игнорирование сопротивления оружия)
105 ref poison applied to weapon (ядовитое оружие)
150 i armor rating (рейтинг брони)
151 i armor type (тип брони)
200 i soul (душа)
201 i capacity (вместимость)
210 bis food (пища))
220 bis poison (яд)

Типы кодов (type codes):
CODE
Тип оружия (weapon type) (103)
0: Blade1H - одноручные мечи
1: Blade2H - двуручные мечи
2: Blunt1H - одноручное дробящее оружие
3: Blunt2H - двуручное дробящее оружие
4: Staff - персональное
5: Bow - лук

CODE
Типы брони (Armor Type) (151)
0: Light Armor - легкая броня
1: Heavy Armor - тяжелая броня

CODE
Уровень души (Soul Level) (200-201)
0: None - нет
1: Petty - крохотная
2: Lesser - небольшая (маленькая)
3: Common - обычная
4: Greater - большая
5: Grand - огромная

9.2 Приложение 2: Слоты ID для экипировки (Slot IDs)

CODE
Броня/одежда (armor/clothing)
0 Голова (head)
1 Волосы (hair)
2 Верхняя часть тела (upper body)
3 Нижняя часть тела (lower body)
4 Руки (hand)
5 Ноги (foot)
6 Правое кольцо (right ring)
7 Левое кольцо (left ring)
8 Амулет (amulet)
13 Щит (shield)
15 Хвост (tail)

Оружие/стрелы (weapon/ammo)
16 Оружие (weapon)
17 Стрелы (боеприпасы (ammo), в Обливионе – стрелы)

Специальные возвращаемые значения, используемые для получения типа экипированных предметов (Value Type 3) при использовании функций GetOV, GetCV, GetEOV и GetECV
CODE
Специальное возвращаемое значение (Special return values):
18 Как верхняя, так и нижняя часть тела (плащи, мантии)

Зарезервированные значения, которые в данный момент не используются и/или устарели
CODE
9 Оружие (weapon)
10 Заплечное оружие (back weapon)
11 Оружие на боку (side weapon)
12 Колчан (quiver)
14 Факел (torch)

9.3 Приложение 3: Сканкоды DX
Функции OBSE работают с аппаратно независимыми кодами DirectX, в частности, модуля DirectInput. В приведенном перечне приведен полный список DX-сканкодов.

CODE
0x01 1 ESCAPE
0x02 2 1
0x03 3 2
0x04 4 3
0x05 5 4
0x06 6 5
0x07 7 6
0x08 8 7
0x09 9 8
0x0A 10 9
0x0B 11 0
0x0C 12 - (“-” минус на главной клавиатуре)
0x0D 13 = (“=” равно)
0x0E 14 backspace
0x0F 15 TAB
0x10 16 Q
0x11 17 W
0x12 18 E
0x13 19 R
0x14 20 T
0x15 21 Y
0x16 22 U
0x17 23 I
0x18 24 O
0x19 25 P
0x1A 26 Left Bracket (левая скобка)
0x1B 27 Right Bracket (правая скобка)
0x1C 28 Enter (Enter на главной клавиатуре)
0x1D 29 Left Control (левый Ctrl)
0x1E 30 A
0x1F 31 S
0x20 32 D
0x21 33 F
0x22 34 G
0x23 35 H
0x24 36 J
0x25 37 K
0x26 38 L
0x27 39 Semicolon (точка с запятой)
0x28 40 Apostrophe (апостроф)
0x29 41 ~ (Console) (“~” тильда, консоль)
0x2A 42 Left Shift (левый Shift)
0x2B 43 Back Slash (обратная косая черта)
0x2C 44 Z
0x2D 45 X
0x2E 46 C
0x2F 47 V
0x30 48 B
0x31 49 N
0x32 50 M
0x33 51 Comma (“,” запятая)
0x34 52 Period (“.” точка на главной клавиатуре)
0x35 53 Slash (“/” косая черта на главной клавиатуре
0x36 54 Right Shift (правый Shift)
0x37 55 Num* (“*” умножение на цифровой клавиатуре)
0x38 56 Left Alt (левый Alt)
0x39 57 Space (пробел)
0x3A 58 Caps Lock
0x3B 59 F1
0x3C 60 F2
0x3D 61 F3
0x3E 62 F4

0x3F 63 F5
0x40 64 F6
0x41 65 F7
0x42 66 F8
0x43 67 F9
0x44 68 F10
0x45 69 Num Lock
0x46 70 Scroll Lock
0x47 71 Num7 (7 на цифровой клавиатуре)
0x48 72 Num8 (8 на цифровой клавиатуре)
0x49 73 Num9 (9 на цифровой клавиатуре)
0x4A 74 Num- (“-“ вычитание на цифровой клавиатуре)
0x4B 75 Num4 (4 на цифровой клавиатуре)
0x4C 76 Num5 (5 на цифровой клавиатуре)
0x4D 77 Num6 (6 на цифровой клавиатуре)
0x4E 78 Num+ (“+” сложение на цифровой клавиатуре)
0x4F 79 Num1 (1 на цифровой клавиатуре)
0x50 80 Num2 (2 на цифровой клавиатуре)
0x51 81 Num3 (3 на цифровой клавиатуре)
0x52 82 Num0 (0 на цифровой клавиатуре)
0x53 83 Decimal (“.” децимальная точка на цифровой клавиатуре)
0x54 84
0x55 85
0x56 86
0x57 87 F11
0x58 88 F12
0x59 89
0x5A 90
0x5B 91
0x5C 92
0x5D 93
0x5E 94
0x5F 95
0x60 96
0x61 97
0x62 98
0x63 99
0x64 100 F13 (NEC PC98)
0x65 101 F14 (NEC PC98)
0x66 102 F15 (NEC PC98)
0x67 103
0x68 104
0x69 105
0x6A 106
0x6B 107
0x6C 108
0x6D 109
0x6E 110
0x6F 111
0x70 112 Kana (Japanese keyboard)
0x71 113
0x72 114
0x73 115
0x74 116
0x75 117
0x76 118
0x77 119
0x78 120
0x79 121 Convert (Japanese keyboard)
0x7A 122
0x7B 123 NoConvert (Japanese keyboard)
0x7C 124
0x7D 125 Yen (Japanese keyboard)
0x7E 126
0x7F 127
0x80 128 Kana (Japanese keyboard)
0x81 129
0x82 130

0x83 131
0x84 132
0x85 133
0x86 134
0x87 135
0x88 136
0x89 137
0x8A 138
0x8B 139
0x8C 140
0x8D 141 NumPadEquals (“=” на цифровой клавиатуре (NEC PC98))
0x8E 142
0x8F 143
0x90 144 CircumFlex (Japanese keyboard)
0x91 145 AT (NEC PC98)
0x92 146 Colon (NEC PC98)
0x93 147 UnderLine (NEC PC98)
0x94 148 Kanji (Japanese keyboard)
0x95 149 Stop (NEC PC98)
0x96 150 AX (Japan AX)
0x97 151 UnLabeled (J3100)
0x98 152 NumPadEnter (Enter на цифровой клавиатуре)
0x99 153
0x9A 154
0x9B 155
0x9C 156 Num Enter
0x9D 157 Right Control (правый Ctrl)
0x9E 158
0x9F 159
0xA0 160
0xA1 161
0xA2 162
0xA3 163
0xA4 164
0xA5 165
0xA6 166
0xA7 167
0xA8 168
0xA9 169
0xAA 170
0xAB 171
0xAC 172
0xAD 173
0xAE 174
0xAF 175
0xB0 176
0xB1 177
0xB2 178
0xB3 179 Num, (“,” запятая на цифровой клавиатуре (NEC PC98))
0xB4 180
0xB5 181 Num/ (“/” косая черта (деление) на цифровой клавиатуре)
0xB6 182
0xB7 183 SysRQ
0xB8 184 Right Alt (правый Alt)
0xB9 185
0xBA 186
0xBB 187
0xBC 188
0xBD 189
0xBE 190
0xBF 191
0xC0 192
0xC1 193
0xC2 194
0xC3 195
0xC4 196
0xC5 197
0xC6 198

0xC7 199 Home (Home on arrow keypad)
0xC8 200 Up Arrow (стрелка вверх)
0xC9 201 PageUp (PgUp on arrow keypad)
0xCA 202
0xCB 203 Left Arrow (стрелка влево)
0xCC 204
0xCD 205 Right Arrow (стрелка вправо)
0xCE 206
0xCF 207 End (End on arrow keypad)
0xD0 208 Down Arrow (стрелка вниз)
0xD1 209 Page Down (PgDn on arrow keypad)
0xD2 210 Insert (Insert on arrow keypad)
0xD3 211 Delete
0xD4 212
0xD5 213
0xD6 214
0xD7 215
0xD8 216
0xD9 217
0xDA 218
0xDB 219 LWin (левая Windows клавиша)
0xDC 220 RWin (правая Windows клавиша)
0xDD 221 AppS (клавиша AppMenu)
0xDE 222
0xDF 223

Альтернативные названия клавиш, облегчающие переход от DOS.
BACKSPACE BACK backspace
NUMPADSTAR MULTIPLY "*" on numeric keypad
LALT LMENU left Alt
CAPSLOCK CAPITAL CapsLock
NUMPADMINUS SUBTRACT "-" on numeric keypad
NUMPADPLUS ADD "+" on numeric keypad
NUMPADPERIOD DECIMAL "." on numeric keypad
NUMPADSLASH DIVIDE "/" on numeric keypad
RALT RMENU right Alt
UPARROW UP UpArrow on arrow keypad
PGUP PRIOR PgUp on arrow keypad
LEFTARROW LEFT LeftArrow on arrow keypad
RIGHTARROW RIGHT RightArrow on arrow keypad
DOWNARROW DOWN DownArrow on arrow keypad
PGDN NEXT PgDn on arrow keypad

9.4 Приложение 4: Игровые ID управления (Control IDs)

Ниже приведена таблица соответствия.
CODE

Cod	Control	Управление
0	Forward	Вперед
1	Back	Назад
2	Slide Left	Шаг влево
3	Slide Right	Шаг вправо
4	Attack	Атака
5	Activate	Активировать
6	Block	Блок
7	Cast	Кастовать
8	Ready Weapon	Вынуть оружие
9	Crouch/Sneak	Красться
10	Run	Бежать
11	Always Run	Всегда бежать
12	Auto Move	Автопередвижение
13	Jump	Прыжок
14	Toggle POV	
15	Menu Mode	Режим меню
16	Rest	Отдых
17	Quick Menu	Быстрое меню
18	Quick1	Быстрый выбор1

19	Quick2	Быстрый выбор2
20	Quick3	Быстрый выбор3
21	Quick4	Быстрый выбор4
22	Quick5	Быстрый выбор5
23	Quick6	Быстрый выбор6
24	Quick7	Быстрый выбор7
25	Quick8	Быстрый выбор8
26	QuickSave	Быстрое сохранение
27	QuickLoad	Быстрая загрузка
28	Grab	Захват

9.5 Приложение 5: Виртуальные сканкоды клавиш, определенные в Windows (Virtual-Key Codes)
Ниже приведены символьные имена констант виртуальных клавиатурных кодов, их значения в десятичном исчислении и их эквиваленты для мыши и клавиатуры, используемые системой.
Коды отсортированы по десятичным кодам в порядке возрастания.

CODE
Symbolic constant name - Value (decimal) - Mouse or keyboard equivalent
VK_LBUTTON 1 Left mouse button
VK_RBUTTON 2 Right mouse button
VK_CANCEL 3 Control-break processing
VK_MBUTTON 4 Middle mouse button (three-button mouse)
VK_XBUTTON1 5 Windows 2000: X1 mouse button
VK_XBUTTON2 6 Windows 2000: X2 mouse button
— 7 Undefined
VK_BACK 8 BACKSPACE key
VK_TAB 9 TAB key
— 10–11 Reserved
VK_CLEAR 12 CLEAR key
VK_RETURN 13 ENTER key
— 14–15 Undefined
VK_SHIFT 16 SHIFT key
VK_CONTROL 17 CTRL key
VK_MENU 18 ALT key
VK_PAUSE 19 PAUSE key
VK_CAPITAL 20 CAPS LOCK key
VK_KANA 21 IME Kana mode
VK_HANGUEL 21 IME Hanguel mode (maintained for compatibility; use VK_HANGUL)
VK_HANGUL 21 IME Hangul mode
— 22 Undefined
VK_JUNJA 23 IME Junja mode
VK_FINAL 24 IME final mode
VK_HANJA 25 IME Hanja mode
VK_KANJI 25 IME Kanji mode
— 26 Undefined
VK_ESCAPE 27 ESC key
VK_CONVERT 28 IME convert
VK_NONCONVERT 29 IME nonconvert
VK_ACCEPT 30 IME accept
VK_MODECHANGE 31 IME mode change request
VK_SPACE 32 SPACEBAR
VK_PRIOR 33 PAGE UP key
VK_NEXT 34 PAGE DOWN key
VK_END 35 END key
VK_HOME 36 HOME key
VK_LEFT 37 LEFT ARROW key
VK_UP 38 UP ARROW key
VK_RIGHT 39 RIGHT ARROW key
VK_DOWN 40 DOWN ARROW key
VK_SELECT 41 SELECT key
VK_PRINT 42 PRINT key
VK_EXECUTE 43 EXECUTE key
VK_SNAPSHOT 44 PRINT SCREEN key
VK_INSERT 45 INS key
VK_DELETE 46 DEL key
VK_HELP 47 HELP key
48 0 key

49 1 key
50 2 key
51 3 key
52 4 key
53 5 key
54 6 key
55 7 key
56 8 key
57 9 key
— 58–64 Undefined
65 A key
66 B key
67 C key
68 D key
69 E key
70 F key
71 G key
72 H key
73 I key
74 J key
75 K key
76 L key
77 M key
78 N key
79 O key
80 P key
81 Q key
82 R key
83 S key
84 T key
85 U key
86 V key
87 W key
88 X key
89 Y key
90 Z key
VK_LWIN 91 Left Windows key (Microsoft® Natural® keyboard)
VK_RWIN 92 Right Windows key (Natural keyboard)
VK_APPS 93 Applications key (Natural keyboard)
— 94 Reserved
VK_SLEEP 95 Computer Sleep key
VK_NUMPAD0 96 Numeric keypad 0 key
VK_NUMPAD1 97 Numeric keypad 1 key
VK_NUMPAD2 98 Numeric keypad 2 key
VK_NUMPAD3 99 Numeric keypad 3 key
VK_NUMPAD4 100 Numeric keypad 4 key
VK_NUMPAD5 101 Numeric keypad 5 key
VK_NUMPAD6 102 Numeric keypad 6 key
VK_NUMPAD7 103 Numeric keypad 7 key
VK_NUMPAD8 104 Numeric keypad 8 key
VK_NUMPAD9 105 Numeric keypad 9 key
VK_MULTIPLY 106 Multiply key
VK_ADD 107 Add key
VK_SEPARATOR 108 Separator key
VK_SUBTRACT 109 Subtract key
VK_DECIMAL 110 Decimal key
VK_DIVIDE 111 Divide key
VK_F1 112 F1 key
VK_F2 113 F2 key
VK_F3 114 F3 key
VK_F4 115 F4 key
VK_F5 116 F5 key
VK_F6 117 F6 key
VK_F7 118 F7 key
VK_F8 119 F8 key
VK_F9 120 F9 key
VK_F10 121 F10 key
VK_F11 122 F11 key

VK_F12 123 F12 key
VK_F13 124 F13 key
VK_F14 125 F14 key
VK_F15 126 F15 key
VK_F16 127 F16 key
VK_F17 128H F17 key
VK_F18 129H F18 key
VK_F19 130H F19 key
VK_F20 131H F20 key
VK_F21 132H F21 key
VK_F22 133H F22 key
VK_F23 134H F23 key
VK_F24 135H F24 key
— 136–143 Unassigned
VK_NUMLOCK 144 NUM LOCK key
VK_SCROLL 145 SCROLL LOCK key
146–150 OEM specific
— 151–159 Unassigned
VK_LSHIFT 160 Left SHIFT key
VK_RSHIFT 161 Right SHIFT key
VK_LCONTROL 162 Left CONTROL key
VK_RCONTROL 163 Right CONTROL key
VK_LMENU 164 Left MENU key
VK_RMENU 165 Right MENU key
VK_BROWSER_BACK 166 Windows 2000: Browser Back key
VK_BROWSER_FORWARD 167 Windows 2000: Browser Forward key
VK_BROWSER_REFRESH 168 Windows 2000: Browser Refresh key
VK_BROWSER_STOP 169 Windows 2000: Browser Stop key
VK_BROWSER_SEARCH 170 Windows 2000: Browser Search key
VK_BROWSER_FAVORITES 171 Windows 2000: Browser Favorites key
VK_BROWSER_HOME 172 Windows 2000: Browser Start and Home key
VK_VOLUME_MUTE 173 Windows 2000: Volume Mute key
VK_VOLUME_DOWN 174 Windows 2000: Volume Down key
VK_VOLUME_UP 175 Windows 2000: Volume Up key
VK_MEDIA_NEXT_TRACK 176 Windows 2000: Next Track key
VK_MEDIA_PREV_TRACK 177 Windows 2000: Previous Track key
VK_MEDIA_STOP 178 Windows 2000: Stop Media key
VK_MEDIA_PLAY_PAUSE 179 Windows 2000: Play/Pause Media key
VK_LAUNCH_MAIL 180 Windows 2000: Start Mail key
VK_LAUNCH_MEDIA_SELECT 181 Windows 2000: Select Media key
VK_LAUNCH_APP1 182 Windows 2000: Start Application 1 key
VK_LAUNCH_APP2 183 Windows 2000: Start Application 2 key
— 184-185 Reserved
VK_OEM_1 186 Windows 2000: For the US standard keyboard, the ';:' key
VK_OEM_PLUS 187 Windows 2000: For any country/region, the '+' key
VK_OEM_COMMA 188 Windows 2000: For any country/region, the ',' key
VK_OEM_MINUS 189 Windows 2000: For any country/region, the '-' key
VK_OEM_PERIOD 190 Windows 2000: For any country/region, the '.' key
VK_OEM_2 191 Windows 2000: For the US standard keyboard, the '/?' key
VK_OEM_3 192 Windows 2000: For the US standard keyboard, the '`~' key
— 193–215 Reserved
— 216–218 Unassigned
VK_OEM_4 219 Windows 2000: For the US standard keyboard, the '[{' key
VK_OEM_5 220 Windows 2000: For the US standard keyboard, the '\|' key
VK_OEM_6 221 Windows 2000: For the US standard keyboard, the ']}' key
VK_OEM_7 222 Windows 2000: For the US standard keyboard, the 'single-quote/double-quote' key
VK_OEM_8 223
— 224 Reserved
225 OEM specific
VK_OEM_102 226 Windows 2000: Either the angle bracket key or the backslash key on the RT 102-key keyboard
227–228 OEM specific
VK_PROCESSKEY 229 Windows 95/98, Windows NT 4.0, Windows 2000: IME PROCESS key
230 OEM specific
VK_PACKET 231 Windows 2000: Used to pass Unicode characters as if they were keystrokes. The VK_PACKET key is the low word of a 32-bit Virtual Key value used for non-keyboard input methods. For more information, see Remark in KEYBDINPUT, SendInput, WM_KEYDOWN, and WM_KEYUP
— 232 Unassigned
233–245 OEM specific

VK_ATTN 246 Attn key
VK_CRSEL 247 CrSel key
VK_EXSEL 248 ExSel key
VK_EREOF 249 Erase EOF key
VK_PLAY 250 Play key
VK_ZOOM 251 Zoom key
VK_NONAME 252 Reserved for future use
VK_PA1 253 PA1 key
VK_OEM_CLEAR 254 Clear key