Справка по написанию скриптов для Проклятых Земель.
v 1.0

Функции языка.

Вместе с этой справкой идет еще 1 файл – ei_script_functions_list. В нем лежит список всех известных нам функций скриптового языка ПЗ, а также их краткое описание. Когда-то давно был просто список имен функций под номерами. То как функции работают – становилось известно при их исследовании различными людьми еще во времена единого Гипата. Поскольку результаты исследования – не документация от Нивала – то описание функций, основанное просто на наблюдении их работы через консоль – может быть неточным или вообще неправильным. Также для некоторых функций описания может не быть вообще – их назначение и принцип работы неизвестны.

Комментарии.

Рекомендую писать комментарии при написании скриптов. Помните, что писать скрипт вы будете только 1 раз. А вот читать его – будет много разных людей и много раз. Написав подробные комментарии вы сильно упростите чтение и понимание скрипта, что ускорит работу со скриптом человеком, который не писал этот скрипт.

Написание комментариев стандартное – 2 слеша, а за ними комментарий. Например:

```
KillScript(  )  //Убить скрипт чтоб больше не выполнялся
GSSetVar(0,"GlobalVarsWasSet",1)  //Этот скрипт больше не выполнится.
Sleep( 2 )
//SendStringEvent( 0, "tutorial DebugMessage" )  //дебаг-мессага
GSSetVar(0,"PlayerIsGod",1)  //Игрок будет гадом
GSSetVar(0,"AllAlly",1)  //Все будут нейтральны
```

Оформление скрипта.

Про скрипты написал. Дальше. Идентификаторы. Рекомендую давать идентификаторам понятные названия. А не что-то вроде Trigger4683246. Пойди потом догадайся что этот номер означает... Не уподобляйтесь Ниваловцам – у них эти номера скорее всего автоматически писал редактор. А мы скрипт редактируем руками.

Про отступы. Форматируйте текст отступами – это сильно увеличит читабельность скрипта. Форматировать по принципу – каждый вложенный блок отступает на 2 пробела больше. Например:

```
Script SetDefVars
//Этот скрипт выставляет умолчальные значение переменных
(
  if
  (
    IsEqual(GSGetVar(0,"GlobalVarsWasSet" ),0)
  )
  then
  (
    KillScript(  )  //Убить скрипт чтоб больше не выполнялся
    GSSetVar(0,"GlobalVarsWasSet",1)  //Этот скрипт больше не выполнится.
    Sleep( 2 )
    //SendStringEvent( 0, "tutorial DebugMessage" )  //дебаг-мессага
    GSSetVar(0,"PlayerIsGod",1)  //Игрок будет гадом
```

```
    GSSetVar(0,"AllAlly",1)  //Все будут нейтральны
  )
)
```

Структура скрипта.

Скрипт состоит из 3 частей (далее в тексте чтобы не было путаницы между понятием “скрипт” - как весь текст скрипта в мобе и понятием “скрипт” как отдельная функция в тексте скрипта – то весь скрипт буду называть “код”. По аналогии и исходным кодом программы чем этот скрипт фактически и является. А словом “скрипт” буду обозначать создаваемые скриптером функции-скрипты.).

1. Объявление глобальных переменных.
2. Объявление скриптов.
3. Реализация скриптов.

Объявление глобальных переменных. По схеме:

```
GlobalVars (
  NULL : object,
  VSS#i#val : object,
  i : object,
  Heroes : group,
  //Пошли объекты на карте
  PlayerIsGodLever : object,
  AllAllyLever : object,
  PlayerIsGodLever_switcher : object,
  AllAllyLever_switcher : object,
  StartSuslaLever : object,
  PlayerIsGodLever_tablicka : object,
  AllAllyLever_tablicka : object,
  StartSuslaLever_tablicka : object,
  TestersLever_tablicka : object,

  //Пошли игровые переменные...
  PlayerIsGod : float,   //Если это 1 то на всех мапах перс будет под гадмодом.
  AllAlly : float        //Если это 1 то на всех мапах все к персу будут нейтральны.
)
```

Тоесть – после слова GlobalVars начинается блок (ограничен скобками). Внутри блока – список переменных и их типов. Тип переменной – то что называют “тип данных” в языках программирования. Тоесть – например если тип переменной string, то в этой переменной может храниться текстовая строка. А если тип float – то число. Итак, известные типы данных в скриптовом языке ПЗ:

object – объект в мобе. Это может быть любой объект на карте – юнит, дерево, камень.... Используя переменную этого типа над объектом можно выполнять различные действия – например убить, переместить, включить, выключить, приказать идти, атаковать и т.д. Перед тем как использовать эту переменную (после объявления она типа пуста) – ей надо присвоить объект. Например так:

```
StartSuslaLever = GetObjectByID( "4640" )  //Рычаг, запускающий Суслу.
```

Здесь переменной StartSuslaLever присваивается объект с номером 4640. Для этого используется функция GetObjectByID, которая возвращает объект с номером, переданным функции аргументом.

group – игровая группа. В игровой группе может быть несколько объектов. Добавить объект в группу можно функцией AddObject.

float – число. Может использоваться как логический тип – 0 соответствует false, 1 соответствует true.

string – строка.

Итак, чтоб объявить переменную – надо написать имя переменной, затем после двоеточия – его тип. Если после этой переменной будут еще переменные объявляться – то надо поставить запятую после имени типа. Если нет – ничего не ставить. Только скобку закрыть не забыть :).

Сразу скажу о присваивании значений переменных и возвращении значений. Используйте функции GSGetVar и GSSetVar. Удалить переменную можно функцией GSDelVar.

Если переменная объявлена как глобальная, и ей присвоено значение – она может быть использована в любом другом мобе, а не только в том где она объявлена. Значение переменной сохраняется – если в одном мобе какой-то переменной было присвоено значение “TEST”, то если в другом мобе просмотреть значение этой переменной – оно так же будет “TEST”.

Объявление скриптов. Происходит в виде:

```
DeclareScript DebugScript (  this : object )
```

После слова DeclareScript пишут имя скрипта, затем в скобках параметры. Параметры объявляют точно так же как объявляют переменные – имя переменной, потом через двоеточие ее тип. Если переменных больше одной – то разделяют их запятыми. Фактически – параметр – это локальная переменная, ее можно использовать только внутри скрипта, параметром которого она является. Переменных можно объявить несколько. Например:

```
DeclareScript #OnBriefingComplete (  nPlayer : float,  szComplete : string )
```

ВАЖНО! В объявлении скрипта ВСЕГДА должен быть параметр this типа object. Приведенный выше в примере скрипт #OnBriefingComplete – это скрипт-исключение, о нем будет отдельный разговор.

После того как скрипт объявлен – его можно использовать как обычную функцию. Например:

```
DebugScript(NULL)
```

Как видно из примера – в качестве значения параметра this надо вписывать NULL. Если параметров больше одного – их значения также пишутся через запятую.

Реализация скриптов. В этом разделе необходимо описать реализацию ВСЕХ объявленных ранее скриптов, а также обязательно должна быть реализация скрипта WorldScript.

Реализация обычных скриптов. Выполняется по схеме:

```
Script ИМЯ_СКРИПТА
(
 if
 (
  УСЛОВИЕ_ВЫПОЛНЕНИЯ
 )
 then
 (
  ТЕЛО_СКРИПТА
 )
 [
 if
 (
  УСЛОВИЕ_ВЫПОЛНЕНИЯ
 )
 then
 (
  ТЕЛО_СКРИПТА
 )
 ]
)
```

То что отмечено квадратными скобками – может повторяться сколько угодно раз, в том числе ноль раз – тоесть содержимого квадратных скобок может и не быть вообще. Сами квадратные скобки естественно не пишут.

ИМЯ_СКРИПТА – имя скрипта, которое вы объявляли в предыдущем разделе. Параметры скрипта здесь не пишут.

УСЛОВИЕ_ВЫПОЛНЕНИЯ – переменная типа float, либо выражение, возвращающее значение типа float. Если значение это равно нулю, то ТЕЛО_СКРИПТА не выполнится. Если равно единице – выполнится.

Фактически ТЕЛО_СКРИПТА – на самом деле тело оператора if. Операторов if в скрипте может быть сколько угодно, но больше нуля :). Выполняются все операторы if, и те, УСЛОВИЕ_ВЫПОЛНЕНИЯ которых равно единице – выполняются.

ТЕЛО СКРИПТА – в этом случае – последовательность строк. В каждой строке вызывается какая-либо функция (функцией может быть как встроенная в игру функция, так и написанный разработчиком скрипт). При выполнении тела скрипта – строки выполняются построчно сверху вниз. Пример:

```
Script PlayerIsGod_CHECK_2
//Вложенный скрипт.
//Вызывается из PlayerIsGod_CHECK
//В соответствии с переменной выставляет гадмод.
(
 if
 (
  IsEqual(GSGetVar(0,"PlayerIsGod"),1)   //Надо быть гадом.
 )
 then
 (
  KillScript( )
  GodMode(0,2)
```

```
  )
  if
  (
    IsEqual(GSGetVar(0,"PlayerIsGod"),0)   //Не гады мы :).
  )
  then
  (
    KillScript( )
    GodMode(0,0)
  )
)
```

В этом примере, как видно больше 1 оператора if. Вообще он может быть только один, например:

```
Script AllAlly_CHECK
//Переключает состояние нейтральности в соотв. с переменной.
(
  if
  (
    IsEqual(GSGetVar(0,"AllAlly" ),Not(GSGetVar(0,"Really_AllAlly")))   //Если переменные несовпадают.
  )
  then
  (
    GSSetVar(0,"Really_AllAlly",GSGetVar(0,"AllAlly"))   //"Совпадить" переменные
    AllAlly_CHECK_2 (NULL)  //Собсно выставить гадмод
  )
)
```

Важная особенность. После того как скрипт вызван, но еще не выполнилось условие ни одного **if** – то скрипт будет “ждать” когда выполнится условик. Если во время работы скрипта у вас не будет вызван ни 1 разу **KillScript( )** - то скрипт даже после выполнения тела **if** будет выполняться. Тоесть – по второму, третьему и т.д. кругу. Поэтому ставить в скриптах **KillScript( )** - это практически всегда обязательно кроме тех случаев когда вы ЧЕТКО понимаете, что делаете (один из таких случаев – предыдущий пример, этот скрипт как раз используется для того, чтобы работать ВСЕГДА и реагировать на изменение переменных).

Еще 1 особенность. Даже если вы убили скрипт командой **KillScript( )** - помните, что при каждом входе в зону заново выполняется WorldScript. Поэтому ваш скрипт вполне может оказаться вызванным опять. С этим связан такой баг в оригинальной игре как бесконечная выдача костей зеленого дракона при входе в Пески после того как Зак обворовал ящеров. Там стоит проверка – если сундук ящеров открыт – то выдать Заку 2 кости. Скрипт после этого убивается (**KillScript( )**). Прикол в том – что при следующем входе в зону – происходит вызов WorldScript, который в том числе вызывает и тот скрипт, который выдает кости. Скрипт, занимающийся костями :D опять проверяет – а не открыт ли сундук. Обнаружив, что сундук открыт – выдает кости. И так до бесконечности. Он то бедный, думает что живет всего 1 раз. А его убивают, а потом опять запускают.... а он все дает и дает кости... В таких случаях обычно ставят дополнительную переменную. Тоесть – надо было бы сделать чтобы после первой выдачи костей выставлялась некая глобальная переменная, сигнализирующая о том что кости уже выданы. А при проверке – проверять не только открыт ли сундук, но еще и состояние этой переменной. Тоесть кости будут выданы только 1 раз. После этого скрипт конечно будет вызываться еще много раз – но уже будет проверять переменную-индиктор и знать что в прошлой жизни уже выдавал игроку кости, и

больше нифига ему не даст. Вот так вот :).

Так. **Теперь о WorldScript.** Это особый скрипт. Он нигде не объявляется. Но именно с него начинается выполнение скриптовой части моба. Именно из этого скрипта происходит вызов всех остальных скриптов. Особый он еще и тем, что в нем нет ни одного оператора if. Команды пишутся прямо в теле скрипта:

```
WorldScript
(
  ТЕЛО_СКРИПТА
)
```

например:

```
WorldScript
(
  //Старт игры.
  Sleep( 10 )  //Чуть подождать чтоб не сглючило...
  SetDefZoneVars_CHECK(NULL)
  //SendStringEvent( 0, "tutorial ForTesters" ) //Выкинуть приветствие тестеру
  Sleep( 20 )
  SetDefVars( NULL )  //Выставить умолчальные значения переменных
  DoNameObjects( NULL ) //Привязать идешники объектов к именным объектам :)
  //Активация проверок...

  //Обычные рычаги
  PlayerIsGodLever_switcher_CHECK(NULL)
  AllAllyLever_switcher_CHECK(NULL)
  StartSuslaLever_CHECK(NULL)

  //Рычаги-индикаторы
  PlayerIsGodLever_CHECK(NULL)
  AllAllyLever_CHECK(NULL)

  //Таблички
  PlayerIsGodLever_tablicka_CHECK(NULL)
  AllAllyLever_tablicka_CHECK(NULL)
  StartSuslaLever_tablicka_CHECK(NULL)
  TestersLever_tablicka_CHECK(NULL)
)
```

Еще 1 особенность – перед именем скрипта не пишется слово Script.

Теперь еще про 1 скрипт-исключение. **#OnBriefingComplete**. Этот скрипт в принципе почти ничем не отличается от обычного скрипта. Он объявляется так же как все скрипты. Он реализуется так же как все скрипты. Но с тремя особенностями:

1. В его параметрах нет параметра this. Скрипт всегда должен объявляться как DeclareScript #OnBriefingComplete (nPlayer : float, szComplete : string).
2. Скрипт не надо вызывать из WorldScript или какого-либо другого скрипта. Этот скрипт вызывается игрой сразу после завершения любого диалога и используется для выполнения действий после диалогов.
3. szComplete – в ней внутри скрипта записана текстовая строка – имя только-что завершенного брифинга. Используется для проверки – какой же брифинг был только-что

завершен. Например:

```
Script #OnBriefingComplete
(
  if
  (
    IsEqualString( szComplete, "b.bz14h.gl63" )
  )
  then
  (
    KillScript(  )
    GSSetVar( 0, "z.gz17h_bz15h", 1 )
  )
)
```

Тоесть это означает – если завершен брифинг b.bz14h.gl63 – то выполнить то что после then.