Уроки скриптологии, часть 1.

Тема первая: Типы переменных.

В скриптах используется всего несколько типов переменных:
void - пустой тип
int - целое число
float - число с плавающей точкой
string - строка текста
class - структура с внутренними типами и функциями
prototype - обычно инициализация класса, но возможно и другое использование
instance - пока сказать нечего.
вот, кажется, и все типы.

Тема вторая: Операции над данными.

Существует несколько типов операций: арифметические, присваивания, отношения, логические, побитовые, над массивами, операции с членами классов.
В основном, все операции применимы только к простым типам, за исключением типа string.

1. Арифметические операции.

\+ - простое сложение
\- - простое вычитание
* - простое умножение
/ - простое деление (результат целая часть)
% - простое деление (результат остаток)

2. Операции присваивания.

= - простое присваивание
+= - сложение с приваиванием
-= - вычитание с присваиванием
*= - умножение с присваиванием
/= - деление с приваиванием (целая часть)

3. Операции отношения.

< - истина, если меньше
> - истина, если больше
<=- истина, если меньше или равно
>= - истина, если больше или равно
== - истина, если равно
!= - истина, если не равно

4. Логические операции.

! - истина, если выражение ложно
|| - истина, если истинно значение первого или второго операнда
&& - истина, если истинно значение первого и второго операнда
Операция ! является только префиксной операцией (пример !операнд)

5. Побитовые операции.

~ - инверсия операнда (префиксная операция)
>> - логический сдвиг вправо операнда1 на величину операнда2
<< - логический сдвиг влево операнда1 на величину операнда2
| - побитовая операция ИЛИ
& - побитовая операция И
^ - побитовая операция исключающее ИЛИ

6. Операции над массивами.

[] - получение или задание элемента массива по индексу (при инициализации массива задание его размерности)

7. Операции с членами классов.

. - обращение к элементу класса
Пример: **аа.бб** , где **аа** - имя класса, **бб** - имя переменной класса

8. Другие операции.

() - вызов функции с аргументами

Тема третья: Операторы и идентификаторы типа.

1. Операторы.

Операторов в скриптах реализовано очень мало:
{ выражение (я) };- оператор **блок**, после закрывающей фигурной скобки обязательна точка с запятой.
Блоки допускают вложения друг в друга.
if (условие(я)) блок - оператор **условного выполнения**. Если условие истина, то выполняется блок, исли условие ложно, то блок пропускается. Возможна и другая реализация:
if (условие(я)) блок else блок - если условие истина, выполняется только блок if, если условие ложь, то выполняется только блок else. Возможно вложение операторов if друг в друга.
return - оператор **выхода из функции**, может возвращать значение.

2. Идентификаторы типа.

Существуют два **идентификатора типа**:
const- определяет тип константы
var - определяет тип переменной
Оба идентификатора типа применимы к следующим типам: int, float, string, func.
Идентификатор var может использоваться только внутри блока.

Тема четвертая: Грамматика скриптов.

Текст скриптов состоит из файлов с расширением **.D**, файлы желательно располагать по определенным темам, согласно директорий, можно создавать любое кол-во файлов и директорий с произвольными именами.
Процесс компиляции начинается с файла с расширением **.SRC**, в нем записывается порядок, в котором будут компилироваться файлы **.D**. Допускается вложенность файлов **.SRC**.

Одно общее правило: прежде чем использовать какую-либо переменную, она должна быть декларирована. Декларация переменной с определенным именем во всех скриптах должна быть одна. Декларация (объявление) может быть с инициализацией или без нее.
Формат **.D** файла прост: любая декларация должна начинаться с одного из ключевых слов: const, var, func, class, instance, prototype. Идентификатор var, как говорилось ранее, может быть использован только в блоке.
Компилятор одинаково распознает как заглавные буквы, так и строчные буквы, поэтому имена Аааа и аааа - это одно и то же.
Комментарии могут быть любые и в любом месте файла, они начинаются с // и до конца строки, или выделяются конструкцией **/* комментарий */** и могуть включать несколько строк.

Существует несколько главных файлов **.SRC**, их имена изменять нельзя. Это в папке **Content - gothic.src** и **fight.src**, а в папке **System - camera.src, menu.src, music.src, sfx.src, visualfx.src и particlefx.src** -из них компилируются соответствующие ***.dat** файлы.
Дальше будем рассматривать базовые классы и функции. Все конкретные классы и функции будут относиться только к скриптам **gothic.src**. Остальные скрипты пока рассматривать не будем, может потом в этом появится необходимость.
Небольшое замечание: В скриптах кол-во локальных переменных в блоке и размерность любого массива ограничены числом 255.

Тема пятая: Классы.

Все классы описаны в файле **.._intern\Classes.d**

1. Класс параметров НПС.

```
const int MAX_CHAPTER    = 5;   //Макс. количество глав в игре (не используется)
const int MAX_MISSIONS   = 5;   //Макс. количество одновременно выполняемых миссий
const int MAX_HITCHANCE  = 5;   //Макс. размерность массива способностей владения видами оружия

class c_npc
{
   var int     id;                         //Однозначный идентификатор, связывающий конкретного НПС с его параметрами
   var string  name[5];                    //Имя НПС, оно появляется на экране, если НПС находится в фокусе
   var string  slot;                       //(не используется) Можно использовать для хранения какой-нибудь строки текста!
   var string  effect;                     //Имя визуального эффекта, привязанного к НПС
   var int     npcType;                    //Тип НПС (описывается в файле ..AI\AI_Intern\AI_Constants.d)
   var int     flags;                      //Флаги НПС (NPC_FLAG_IMMORTAL, NPC_FLAG_GHOST)
   var int     attribute[ATR_INDEX_MAX];   //Массив значений атрибутов (ATR_HITPOINTS, ATR_MANA, ATR_STRENGTH и т.д.)
   var int     HitChance[MAX_HITCHANCE];   //Способность владения НПС определенным оружием
   var int     protection[PROT_INDEX_MAX]; //Массив значений защиты от определенных повреждений
   var int     damage[DAM_INDEX_MAX];      //Массив значений наносимых повреждений
   var int     damagetype;                 //Тип повреждения (НПС наносит в данный момент)
   var int     guild;                      //Принадлежность к гильдии
   var int     level;                      //Уровень НПС
   var func    mission[MAX_MISSIONS];      //(не используется)
   var int     fight_tactic;               //Тактика сражения (описывается в файле ..AI\AI_Intern\AI_Constants.d)
   var int     weapon;                     //(не используется)
   var int     voice;                      //Номер голоса в диалогах
   var int     voicePitch;                 //Тональность голоса
   var int     bodymass;                   //(не используется)
   var func    daily_routine;              //Функция распорядка дня
   var func    start_aistate;              //Функция начального AI состояния (ZS_ функции)
   var string  spawnPoint;                 //Имя Waypoint, в котором рождается НПС
   var int     spawnDelay;                 //Задержка в секундах между смертью и возрождением НПС
   var int     senses;                     //Чувства (зрение, слух, обоняние)
   var int     senses_range;               //Диапазон чувств в см.
   var int     aivar[100];                 //Массив значений AI переменных (описывается в файле ..AI\AI_Intern\AI_Constants.d), неиспол
 использоваться для собственных целей.
   var string  wp;                         //Текущий Waypoint, в котором находится НПС (отслеживается Gothic.exe)
   var int     exp;                        //Количество полученной экспы (только для ГГ)
   var int     exp_next;                   //Количество экспы оставшееся до следующего уровня (только для ГГ)
   var int     lp;                         //Свободные очки обучения (только для ГГ)
   var int     bodyStateInterruptableOverride; //Флаг возможности прерывания действия НПС (если TRUE - НПС прервать нельзя)
   var int     noFocus;                    //Флаг, отвечающий за фокус (если TRUE и НПС находится в фокусе, то у НПС не показывается им
};
```

2. Класс миссий (не используется).

```
class c_mission
{
   var string  name;
   var string  description;
   var int     duration;
```

```
   var int     important;
   var func    offerConditions;
   var func    offer;
   var func    successConditions;
   var func    success;
   var func    failureConditions;
   var func    failure;
   var func    obsoleteConditions;
   var func    obsolete;
   var func    running;
};
```

3. Класс параметров предмета.

```
class c_item
{
   var int     id;                         //Однозначный идентификатор, связывающий конкретный предмет с его параметрами
   var string  name;                       //Имя предмета, оно появляется на экране, если предмет находится в фокусе
   var string  nameID;                     //(не используется) Можно использовать для хранения какой-нибудь строки текста!
   var int     hp,hp_max;                  //???
   var int     mainflag;                   //Флаги категории предмета
   var int     flags;                      //Остальные флаги предметов
   var int     weight,value;               //Вес и стоимость предмета
   var int     damageType;                 //Тип наносимого повреждения (используется для оружия)
   var int     damageTotal;                //Суммарное повреждение (используется для оружия)
   var int     damage[DAM_INDEX_MAX];      //Величина ущерба по типам повреждений (используется для оружия)
   var int     wear;                       //Флаги ношения предмета
   var int     protection[PROT_INDEX_MAX]; //Величина защиты от конкретных повреждений (используется для брони)
   var int     nutrition;                  //(не используется)
   var int     cond_atr[3];                //Массив атрибутов, требуемых для применения
   var int     cond_value[3];              //Массив значений атрибутов, требуемых для применения
   var int     change_atr[3];              //(не используется)
   var int     change_value[3];            //(не используется)
   var func    magic;                      //(не используется)
   var func    on_equip;                   //Функция вызываемая при экипировке
   var func    on_unequip;                 //Функция вызывается, когда предмет снимается или убирается в инвентарь
   var func    on_state[4];                //Функции вызываемые при использовании предмета по видам состояний (видов состояний не обнар
   var func    owner;                      //Владелец предмета, записывается ID НПС (не понятен тип переменной FUNC ???)
   var int     ownerGuild;                 //Какой гильдии принадлежит предмет
   var int     disguiseGuild;              //(не используется)
   var string  visual;                     //Имя .3DS файла текстур предмета
   var string  visual_change;              //Имя .ASC файла визуальных эффектов при получении этого предмета
   var string  effect;                     //Имя визуального эффекта, привязанного к предмету
   var int     visual_skin;                //Вариация текстуры для .3DS файла текстур предмета
   var string  scemeName;                  //Внутреннее имя предмета
   var int     material;                   //Материал предмета
   var int     munition;                   //Тип заряда для оружия дальнего радиуса поражения
   var int     spell;                      //Номер заклинания, которое вызывает данный предмет
   var int     range;                      //Радиус поражения оружием ближнего боя
   var int     mag_circle;                 //Круг магии, к которому принадлежит предмет
   var string  description;                //Название предмета в инвентаре
   var string  text[ITM_TEXT_MAX];         //Массив строк описания предмета в инвентаре
   var int     count[ITM_TEXT_MAX];        //Массив значений характеристик предмета в инвентаре (соответствует определенной текстовой с
   // Характеристики изображения предмета в инвентаре
   var int     inv_zbias;                  //Масштаб предмета в инвентаре
   var int     inv_rotx;                   //Поворот предмета в инвентаре относительно оси X
   var int     inv_roty;                   //Поворот предмета в инвентаре относительно оси Y
   var int     inv_rotz;                   //Поворот предмета в инвентаре относительно оси Z
   var int     inv_animate;                //Вращение предмета в инвентаре
};
```

4. Класс параметров фокуса.

```
class c_focus
{
   // для НПС
   var float   npc_longrange;              //Ширина
   var float   npc_range1,npc_range2;      //Диапазон дальности
   var float   npc_azi;                    //Азимут
   var float   npc_elevdo,npc_elevup;      //Тангаж
   var int     npc_prio;                   //Приоритет
   // для предметов
   var float   item_range1,item_range2;    //Диапазон дальности
   var float   item_azi;                   //Азимут
   var float   item_elevdo,item_elevup;    //Тангаж
   var int     item_prio;                  //Приоритет
   // для MOB's
   var float   mob_range1,mob_range2;      //Диапазон дальности
   var float   mob_azi;                    //Азимут
   var float   mob_elevdo,mob_elevup;      //Тангаж
   var int     mob_prio;                   //Приоритет
};
```

5. Класс параметров диалогов.

```
class c_info
{
   var int     npc;                        //ID НПС, которому принадлежит диалог
   var int     nr;                         //Опция диалога с большим nr будет отображаться в списке ниже
   var int     important;                  //Флаг начала диалога (TRUE - диалог начинаешь ты, FALSE - с тобой)
   var func    condition;                  //Функция проверки условий начала диалога (пока функция не возвратит TRUE, диалог не начнетс
   var func    information;                //Функция управления диалогом
   var string  description;                //Строка, которая выводится в окно выбора вариантов ответов
   var int     trade;                      //Возможность торговли (TRUE - возможно торговать, FALSE - нет)
   var int     permanent;                  //Постоянность диалога (TRUE - диалог повторяется при каждом обращении, FALSE - однократный)
};
```

6. Класс реакций на предметы (не используется).

```
class c_itemreact
{
   var int     npc;
   var int     trade_item;
   var int     trade_amount;
```

```
   var int     requested_cat;
   var int     requested_item;
   var int     requested_amount;
   var func    reaction;
};
```

7. Класс параметров заклинаний.

```
class c_spell
{
   var float   time_per_mana;               //Время в мс, требуемое для инвестирования маны на один уровень
   var int     damage_per_level;            //Наносимый урон за уровень заклинания
   var int     damageType;                  //Тип повреждения
   var int     spellType;                   //Категория заклинания
   var int     canTurnDuringInvest;         //Возможность поворота во время инвестирования маны (TRUE - возможно)
   var int     canChangeTargetDuringInvest; //Возможность выбора цели во время инвестирования маны (TRUE - возможно)
   var int     isMultiEffect;               //Возможность мультиэффектов заклинания (множественные цели и др.)
   var int     targetCollectAlgo;           //Константа описания цели
   var int     targetCollectType;           //Тип цели заклинания
   var int     targetCollectRange;          //Дальность действия
   var int     targetCollectAzi;            //Угол азимута
   var int     targetCollectElev;           //Угол тангажа
};
```

8. Глобальные ссылки на классы.

```
instance self,other(c_npc);  // Сам НПС и Другой НПС
instance victim(c_npc);      // НПС - жертва определенного действия (Зий в действии)
instance item(c_item);       // Сам предмет
instance hero(c_npc);        // Сам ГГ (игрок)
```

Тема шестая: Функции.

Несколько общих замечаний:
1. Функции могут возвращать значения только этих типов: void, int, float, string, instance.
2. Рассматривать будем пока только базовые функции, которые реализованы в екзешнике.
3. Большинство базовых функций описано в файле **externals.d**, но в нем есть ошибки (этот файл приведен только для справки, он не компилируется), о них буду говорить по мере разбора функций, и отмечать значком ***.
4. Идентификаторы func (при объявлении функции) и var (как идентификатор аргументов) для простоты приводить не буду.

1. Функции вывода текста.

void Print (string s0); - выводит на экран строку текста s0, как OutputUnits текст (аналогично выводу из OU файла).
void PrintMulti (string s0, string s1, string s2, string s3, string s4); - выводит текст на экран, объединяя строки параметров s0- s 4 в одну строку.
int PrintScreen (string msg, int posx, int posy, string font, int timeSec); - выводит на экран строку текста msg (имя шрифта – font) с координатами posx, posy (диапазон от 0 до 99% размера экрана, -1 означает вывод по центру соответствующей оси экрана) на время timeSec (в секундах). Возвращаемое значение всегда = 0.***
void PrintDebug (string s); - выводит текст строки s через zSpy или в лог. файл, работает только в отладочном режиме (включается (Alt+D))
void PrintDebugInst (string text); - выводит текст строки text через zSpy или в лог. файл для ссылки self в отладочном режиме.
void PrintDebugInstCh (int ch, string text); - выводит текст строки text через zSpy или в лог. Файл для ссылки self канала ch в отладочном режиме (каналы определены в файле PrintDebug. d).
void PrintDebugCh (int ch, string text); - выводит текст строки text через zSpy или в лог. файл для канала ch в отладочном режиме
int PrintDialog (int dialogNr, string msg, int posx, int posy, string font, int timeSec); - работает аналогично функции PrintScreen, только вывод осуществляется в окно диалога с номером dialogNr (координаты в % действуют внутри окна диалога).***

2. Функции преобразования типов.

string IntToString (int x); - преобразовывает целое число x в строку.
string floatToString (float x); - преобразовывает число с плавающей точкой x в строку.***
int floatToInt (float x); - преобразовывает число с плавающей точкой x в целое число.
float IntTofloat (int x); - преобразовывает целое число x в число с плавающей точкой.
string ConcatStrings (string str1, string str2); - результирующей строкой является объединение двух строк str1 и str2.

3. Функции работы со звуком.

void Snd_Play (string s0); - воспроизвести звуковой файл с именем s0.
void Snd_Play3D (c_npc n0, string s1); - воспроизвести звуковой файл с именем s0 для НПС n0.
int Snd_IsSourceNpc (c_npc self); - проверить, является ли НПС self источником звука (return > 0) и установить ссылку other на этого НПС.
int Snd_IsSourceItem (c_npc self); - проверить, является ли у НПС self предмет источником звука (return > 0) и установить ссылку item на этот предмет
int Snd_GetDistToSource (c_npc self); - выдать расстояние в сантиметрах до источника звука от НПС self.

4. Вспомогательные функции.

int Hlp_Random (int n0); - сгенерировать случайное целое число в диапазоне от 0 до n0 - 1.
int Hlp_StrCmp (string s1, string s2); - сравнивает две строки s1 и s2, возвращает TRUE при равенстве строк или FALSE при неравенстве.
int Hlp_IsValidNpc (c_npc self); - если ссылка на НПС self существует, то возвращает TRUE, иначе – FALSE.
int Hlp_IsValidItem (c_item item); - если ссылка на предмет item существует, то возвращает TRUE, иначе – FALSE.
int Hlp_IsItem (c_item item, int instanceName); - проверяет принадлежность предмета item к соответствующему типу instanceName, возвращает TRUE при соответствии, иначе – FALSE.
c_npc Hlp_GetNpc (int instanceName); - возвращает ссылку на НПС, принадлежащему к типу instanceName.
int Hlp_GetInstanceID (c_npc npc); - получить ID номер npc по ссылке на класс npc.
int Hlp_GetInstanceID (c_item item); - получить ID номер предмета по ссылке на класс item.

5. Функции работы с документами (свитки, книги, записки, карты).

int Doc_Create (); - создает новый документ и возвращает номер его обработчика.
int Doc_CreateMap (); - создает новую карту и возвращает номер её обработчика. ***

void Doc_SetLevel (int handle, string world); - устанавливает уровень world (формат ZEN) для документа handle.*** (Не знаю, что конкретно делает эта функция).
void Doc_SetPages (int handle, int pages); - задает кол-во отображаемых страниц pages для документа handle.
void Doc_SetPage (int handle, int page, string pageimage, int scale); - устанавливает параметры страницы page (если параметр page = -1, то применяется ко всем страницам) документа handle: pageimage – имя файла изображения страницы в формате TGA, scale = 0 - применять масштабируемось, scale = 1 - не использовать масштабируемость (применяется для карт).
void Doc_SetFont (int handle, int page, string fontname); - задать шрифт для страницы page документа handle, где fontname – имя файла шрифта в формате TGA.
void Doc_SetMargins (int handle, int page, int leftMargin, int topMargin, int rightMargin, int bottomMargin, int pixel); - установить границы вывода текста на странице page документа handle, где leftMargin, topMargin, rightMargin, bottomMargin – соответственно левая, верхняя, правая и нижняя границы, pixel – кол-во пикселей на единицу границы.
void Doc_PrintLine (int handle, int page, string text); - выводит строку текста text на странице page документа handle.
void Doc_PrintLines (int handle, int page, string text); - выводит текст text в несколько строк на странице page документа handle.
void Doc_Show (int handle); - выводит созданный документ handle на экран.
void Doc_Open (string picName); - открывает документ с именем picName, как рисунок на заднем плане (в скриптах не используется).
void Doc_Font (string fontName); - задает для документа шрифт fontName (в скриптах не используется).
void Doc_Print (string text); - выводит текст text документа на экран (в скриптах не используется).
void Doc_MapCoordinates (string levelName, float gamex1, float gamey1, float pixelx1, float pixely1, float gamex2, float gamey2, float pixelx2, float pixely2); - инициализирует карту с именем уровня levelName, где gamex1, gamey1, gamex2, gamey2 – координаты углов карты, соответствующие координатам уровня, а pixelx1, pixely1, pixelx2, pixely2 – соответственно коодинаты пикселей углов карты (в скриптах не используется).

6. Функции работы с журналом квестов.

void Log_CreateTopic (string name, int section); - создать в журнале тему с именем name в разделе section (предопределены два раздела LOG_MISSION или LOG_NOTE).
void Log_SetTopicStatus (string name, int status); - установить статус status темы с именем name (предопределены следующие статусы - LOG_RUNNING, LOG_SUCCESS, LOG_FAILED, LOG_OBSOLETE).
void Log_AddEntry (string topic, string entry); - записать в тему topic текст из строки entry.

7. Функции работы с миссиями.

void Mis_SetStatus (int missionName, int newStatus); - установить статус newStatus миссии missionName (предопределены следующие статусы RUNNING, SUCCESS, FAILED и т.д.)
int Mis_GetStatus (int missionName); - получить статус миссии missionName.
int Mis_OnTime (int missionName); - проверка миссии, возвращает TRUE, если миссия активна, т.е. не завершена. Примечание: данная функция в экзешнике реализована неверно, она ничего не возвращает (лучше ее не использовать).
void Mis_AddMissionEntry (c_mission miss, string name); - добавить в список миссий новую миссию miss под именем name.***
void Mis_RemoveMission (c_mission miss); - удалить из списка миссий миссию miss. ***

8. Функции работы с инвентарем.

void CreateInvItem (c_npc n0, int n1); - создает в инвентаре НПС n0 предмет с ID n1.
void CreateInvItems (c_npc n0, int n1, int n2); - создает в инвентаре НПС n0 предметы с ID n1 количеством n2.
void EquipItem (c_npc n0, int n1); - экипировать НПС n0 предметом с ID n1.
void Mob_CreateItems (string mobName, int itemInstance, int amount); - создает в контейнере с именем mobName (например сундук) предметы с ID itemInstance количеством amount.
int Mob_HasItems (string mobName, int itemInstance); - получить количество предметов с ID itemInstance, находящихся в контейнере с именем mobName.

9. Функции анимации.

void Mdl_ApplyOverlayMds (c_npc n0, string s1); - запустить файл анимации s1 с расширением mds для НПС n0.
void Mdl_RemoveOverlayMDS (c_npc self, string overlayName); - отключить от НПС self файл анимации overlayName.
void Mdl_ApplyOverlayMDSTimed (c_npc self, string overlayname, int timeTicks); - запустить файл анимации overlayname для НПС self на время timeTicks.***
void Mdl_ApplyRandomAni (c_npc n0, string s1, string s2); - запустить случайные анимации s1 для НПС n0 в последовательности анимационных состояний s2.
void Mdl_ApplyRandomAniFreq (c_npc n0, string s1, float f2); - устанавливает частоту выборки f2 случайных анимационных состояний s1 для НПС n0.
void Mdl_StartFaceAni (c_npc self, string name, float intensity, float holdTime); - запускает файл name анимации лица для НПС self с интенсивностью intensity (в %, 1 – 100%) и временем удержания holdTime (1 – постоянно).
void Mdl_ApplyRandomFaceAni (c_npc self, string name, float timeMin, float timeMinVar, float timeMax, float timeMaxVar, float probMin); - запускает файл случайной анимации лица name для НПС self со следующими параметрами : timeMin, timeMax – диапазон времени воспроизведения анимации в сек, timeMinVar, timeMaxVar – отклонения от значений timeMin, timeMax в сек, probMin – вероятность использования отклонения (от 0 до 1) от нижней до верхней границы.
void Mdl_SetVisual (c_npc self, string fileMds); - задать для НПС self файл изображения с именем fileMds (формат MDS).***
void Mdl_SetVisualBody (c_npc self, string body, int bodytex, int color, string head, int htex, int ttex, int armor); - установить параметры изображения НПС self, где : body – имя mesh файла изображения тела, bodytex – номер текстуры тела, color – номер палитры цвета кожи, head – имя MMS файла изображения головы, htex – номер текстуры головы, ttex – номер текстуры зубов, armor – номер одетой брони (- 1 – брони нет).***
void Mdl_SetModelScale (c_npc self, float x, float y, float z); - установить масштаб mesh изображения модели НПС self по координатам x, y, z в % (1 -100%).
void Mdl_SetModelFatness (c_npc self, float fatness); - установить жирность mesh изображения модели НПС self в % (1 - 100%). Может кто-нибудь пояснить, что это за функция?

10. Функции распорядка дня.

void TA_Min (c_npc self, int start_h, int start_m, int stop_h, int stop_m, func state, string waypoint); - для НПС self задается функция распорядка дня state, где start_h и start_m – время в часах и минутах начала выполнения функции, stop_h и stop_m – соответственно время окончания функции, waypoint – имя точки, в которой выполняется функция.
void TA (c_npc self, int start_h, int stop_h, func state, string waypoint); - выполняет все то, что и предыдущая функция, только без привязки к минутам.
void TA_CS (c_npc self, string csName, string roleName); - функция, выполняемая последней в списке функций распорядка дня для НПС self, где csName – имя файла CS (Cutscene), roleName – роль НПС, которая должна выполняться. (частично не понял)
void TA_BeginOverlay (c_npc self); - начать выполнять заявленный распорядок дня для НПС self.
void TA_EndOverlay (c_npc self); - закончить выполнять заявленный распорядок дня для НПС self.
void TA_RemoveOverlay (c_npc self); - удалить заявленный распорядок дня для НПС self.

11. Функции работы с диалогами.

int InfoManager_HasFinished (); - показывает, завершен ли диалог (возвращает 0 во время диалога и 1 после окончания диалога).***
void Info_AddChoice (c_info menu, string text, func fn); - добавляет в меню диалога menu строку выбора действия text, fn – функция, которая активируется при выборе данного пункта меню.***
void Info_ClearChoices (c_info menu); - удалить тему диалога menu.***

12. Функции различного назначения.

void ExitGame (); - просто закончить игру.***
int PlayVideo (string video); - показать видео файл video (с расширением BIK), возвращает 1, если видео показано, 0 – в случае ошибки.***
void SetPercentDone (int perc); - установить процент perc отображения полоски в окне загрузки (ProgressBar), значение от 0 до 100.***
void IntroduceChapter (string chapter, string name, string file, string sound, int time); - вывести на экран окно новой главы, где chapter – номер главы, name – название главы, file – файл заставки главы (формат TGA), sound – звуковой файл (формат WAV), time – время показа заставки.***
void Tal_Configure (int tal, int value); - переопределить значение таланта value для константы tal (например, определены константы NPC_TALENT_PICKLOCK, NPC_TALENT_MAGE и т.д.).***
void Perc_SetRange (int percID, int range); - устанавливает дальность действия range пассивного восприятия percID в сантиметрах.
void Rtn_Exchange (string oldRoutine, string newRoutine); - заменяет функции oldRoutine распорядка дня НПС self на новые функции newRoutine. (Имя функции должно начинаться с RTN_ и заканчиваться идентификатором НПС скрипта).
int Hlp_CutscenePlayed (string csName); - информирует, игралась ли Cutscene с именем csName (0 – нет, 1 – да).

13. Функции работы с миром Готики.

int Wld_DetectNpc (c_npc self, int instance, func aiState, int guild); - эта функция инициализирует глобальную переменную скриптов other, отличную от НПС self, где instance – производная от класса c_npc, которая должна быть найдена и проинициализирована (-1 – любая производная), guild – гильдия, членом которой должен быть искомый НПС (-1 – любая гильдия), aiState – функция AI состояния, в котором должен находится искомый НПС (NOFUNC – любое AI состояние). Функция возвращает 1 в случае успешного завершения (other инициализирован найденным НПС), 0 – неудача (other не определен).
int Wld_DetectNpcEx (c_npc self, int instance, func aiState, int guild, int detectPlayer); - выполняет все тоже, что и предыдущая функция, дополнительный параметр detectPlayer указывает, исключить ли игрока (ГГ) из поиска (0 – исключить, 1 – нет).
int Wld_DetectItem (c_npc self, int flags); - эта функция инициализирует глобальную переменную скриптов item предметом, возможно находящимся у НПС self и имеющим флаг flags, возвращает 1 при успешном поиске и инициализации, иначе – 0.
int Wld_DetectPlayer (c_npc self); - возвращает 1, если ГГ есть НПС self, иначе – 0.
void Wld_SetGuildAttitude (int guild1, int attitude, int guild2); - установить отношение между гильдиями, где guild1, guild2 – гильдии, между которыми устанавливается отношение, attitude – отношение между гильдиями (определены отношения ATT_HOSTILE, ATT_FRIENDLY, ATT_NEUTRAL, ATT_ANGRY).
int Wld_GetGuildAttitude (int guild1, int guild2); - получить отношение между гильдиями guild1 и guild2.
int Wld_IsMobAvailable (c_npc self, string schemeName); - проверяет, есть ли МОВ с именем schemeName в окрестности 10 метров от НПС self, возвращает TRUE, если МОВ существует и свободен, иначе FALSE.
int Wld_GetMobState (c_npc self, string schemeName); - определяет состояние МОВ с именем schemeName для НПС self, возвращает состояние или -1, если МОВ не найден.
int Wld_IsFPAvailable (c_npc self, string fpName); - проверяет, есть ли FP (Freepoint) с именем fpName в окрестности 20 метров от НПС self, возвращает TRUE, если FP существует и свободна, иначе FALSE.
int Wld_IsNextFPAvailable (c_npc self, string fpName); - работает аналогично предыдущей функции, но проверится ближайшая точка, если fpName блокирована.
int Wld_GetDay (); - получить текущий день, отсчет дней идет от дня старта (Gamestart) = 0.
int Wld_IsTime (int hour1, int min1, int hour2, int min2); - возвращает 1, если текущее время находится между границами (hour1, min1 – нижняя граница, hour2, min2 – верхняя граница в часах и минутах), иначе возвращает 0.
void Wld_SetTime (int hour, int min); - установить текущее время в часах hour и минутах min.***
void Wld_InsertNpc (int npcInstance, string spawnPoint); - разместить в мире одного НПС, где npcInstance – ссылка на НПС, spawnPoint – имя точки размещения (может быть как WP, так и FP).
void Wld_InsertNpcAndRespawn (int instance, string spawnPoint, int spawnDelay); - выполняется аналогично предыдущей функции, только НПС после смерти будет оживлен в заданной точке через spawnDelay секунд.***
void Wld_SpawnNpcRange (c_npc self, int npcInstance, int number, float time); - НПС self около себя создает НПС (ссылка на него npcInstance) количеством number на время жизни time. Используется в заклинаниях вызова.***
void Wld_RemoveNpc (int npcInstance); - удалить из мира НПС (ссылке на него npcInstance).***
void Wld_InsertItem (int itemInstance, string spawnPoint); - разместить в мире один предмет, где itemInstance – ссылка на предмет, spawnPoint – имя точки размещения (может быть как WP, так и FP). В примечании к функции написано: «Осторожно, не функционирует! Предмет будет вставлен в центр мира.» В скриптах данная функция используется, осталось выяснить, как она работает.
int Wld_RemoveItem (c_item item); - удалить из мира предмет по item ссылке на него, возвращает 1 – успешное удаление, иначе – 0.
void Wld_InsertObject (string name, string point); - разместить в мире визуальный объект name в точке с именем point (может быть как WP, так и FP).***
void Wld_SetObjectRoutine (int hour1, int min1, string objName, int state); -объект с именем objName (VOB объект) перейдет в состояние state в момент времени hour1, min1 (часы и минуты).
void Wld_SetMobRoutine (int hour1, int min1, string objName, int state); - выполняется как и предыдущая функция, только для МОВ объекта.
void Wld_SendTrigger (string name); - активировать функцию VOB объекта с именем name. (Я не знаю где где прописаны эти функции).***
void Wld_SendUntrigger (string name); - деактивировать функцию VOB объекта с именем name.***
void Wld_ExchangeGuildAttitudes (string name); - установить взаимоотношения между гильдиями, где name – имя таблицы заданных отношений.
void Wld_AssignRoomToGuild (string s0, int guild); - размещает гильдию guild на пространстве s0.
void Wld_AssignRoomToNpc (string s0, c_npc roomowner); - размещает НПС roomowner на пространстве s0.
c_npc Wld_GetPlayerPortalOwner ();
int Wld_GetPlayerPortalGuild ();
c_npc Wld_GetFormerPlayerPortalOwner ();
int Wld_GetFormerPlayerPortalGuild ();
Какие-то четыре функции, определяющие положение игрока в пространстве, большего сказать, увы, не могу.
void Wld_PlayEffect (string ???, instance ???, instance ???, int ???, int ???, int ???); - данная функция абсолютно нигде не применяется и не описана, поэтому ее действие и аргументы мне не известны.***

14. Функции работы с НПС.

int Npc_GetTRUEGuild (c_npc npc); - возвращает номер гильдии, к которой принадлежит НПС npc.
int Npc_SetTRUEGuild (c_npc npc, int guildID); - установить принадлежность НПС npc к гильдии guildID.
void Npc_SetAttitude (c_npc self, int att); - установить постояное отношение att НПС self ко всем остальным НПС.
void Npc_SetTempAttitude (c_npc self, int att); - установить временное отношение att НПС self ко всем остальным НПС.
int Npc_GetAttitude (c_npc self, c_npc other); - получить любое отношение НПС self к НПС other.
int Npc_GetPermAttitude (c_npc self, c_npc other); - получить постоянное отношение НПС self к НПС other.
int Npc_GetGuildAttitude (c_npc self, c_npc other); - получить отношение НПС self к НПС other, как отношение между гильдиями, а не личностями.
void Npc_SetKnowsPlayer (c_npc self, c_npc player) ; - эта функция задает, что НПС self знаком с НПС player (ГГ).
int Npc_KnowsPlayer (c_npc self, c_npc player); - проверить, знаком ли НПС self с НПС player, возвращает TRUE, если знаком, иначе FALSE.
int Npc_OwnedByNpc (c_item item, c_npc npc); - проверить, принадлежит ли предмет item npc, возвращает 1, если принадлежит, иначе - 0.
int Npc_OwnedByGuild (c_item item, int guild); - - проверить, принадлежит ли предмет item гильдии guild, возвращает 1, если принадлежит, иначе - 0.

string Npc_GetNearestWP (c_npc self); - возвращает название WP, в которой расположен НПС self.
string Npc_GetNextWP (c_npc self); - возвращает название соседней точки WP, рядом с которой расположен НПС self.
int Npc_IsWayBlocked (c_npc self); - возвращает 1, если путь НПС self прегражден препятствием, иначе – 0.
int Npc_IsOnFP (c_npc self, string name); - проверяет, находится ли НПС self в точке FP с именем name, возвращает 1 – да, иначе – 0.
int Npc_IsDead (c_npc n0); - возвращает 1, если НПС n0 мертв, иначе – 0.
int Npc_KnowsInfo (c_npc self, int infoInstance); - возвращает TRUE, если НПС self уже получил информацию infoInstance, иначе FALSE.
int Npc_CheckInfo (c_npc npc, int important); - проверяет, имеет ли НПС npc действительную информацию для игрока, в этом случае возвращается 1 и начинается диалог, иначе 0, где important – важность информации (1 – важная, 0 – обычная).
int Npc_GiveInfo (c_npc npc, int important); - работает аналогично предыдущей функции, есть какое-то отличие, но я его не понял.
int Npc_CheckAvailableMission (c_npc npc, int missionState, int important); - проверяет, имеет ли npc миссию в состоянии missionState (AVAILABLE, RUNNING) с важностью important, возвращает 1, если имеет, иначе – 0.
int Npc_CheckRunningMission (c_npc npc, int important); - работает аналогично предыдущей функции, только проверяется миссия в состоянии текущего выполнения.
int Npc_CheckOfferMission (c_npc npc, int important); - работает аналогично предыдущей функции, только проверяется миссия в которой НПС npc может что-то предложить игроку.
int Npc_GetStateTime (c_npc self); - возвращает кол-во секунд, которые НПС self находится в текущем состоянии, заданном в " Loop " цикле.
void Npc_SetStateTime (c_npc self, int seconds); - установить кол-во секунд seconds, как долго НПС self может находится в этом состоянии.
int Npc_GetBodyState (c_npc self); - возвращает состояние, в котором находится НПС self (BS_константы).
int Npc_HasBodyFlag (c_npc self, int bodyFlag); - проверяет, установлен ли у НПС self флаг состояния bodyFlag, возвращает 1, если установлен, иначе – 0.
int Npc_IsPlayer (c_npc player); - возвращает 1, если проверяемый НПС player является ГГ.
int Npc_HasDetectedNpc (c_npc self, c_npc other); - возвращает 1, если НПС self чувствует (видит, сышит и т.д.) НПС other, иначе – 0.
int Npc_IsInState (c_npc self, func state); - запрос на текущее состояние фигуры НПС self, где state – функция состояния фигуры, возвращает TRUE, если НПС находится в этом состоянии, иначе – FALSE.
int WasInState (c_npc self, func state); - аналогично предыдущей функции, только запрос на предыдущее состояние фигуры НПС.
int Npc_IsInRoutine (c_npc self, func state); - проверяет, находится ли НПС self в функции state, возвращает TRUE если находится, иначе – FALSE.
void Npc_ExchangeRoutine (c_npc self, string routineName); - поменять у НПС self распорядок дня routineName.
int Npc_IsInCutscene (c_npc self); - возвращает 1, если НПС self находится во время проигрывания Cutscene, иначе – 0.
int Npc_IsVoiceActive (c_npc self); - возвращает 1, если НПС self разговаривает, иначе – 0.***
void Npc_ClearAIQueue (c_npc self); - удаляет все команды для НПС self из очереди AI_Queue.
void Npc_PlayAni (c_npc self, string file); - воспроизвести файл анимации file для НПС self.***
void Npc_SetRefuseTalk (c_npc self, int timeSec); - установить счетчик отказа от диалога для НПС self на timeSec секунд.
int Npc_RefuseTalk (c_npc self); - проверить, истек ли счетчик отказа от диалога для НПС self, возвращает TRUE, если счетчик не истек, иначе – FALSE.
int Npc_IsNear (c_npc self, c_npc other); - возвращает 1, если НПС other находится на расстоянии не менее 3 метров от НПС self, иначе – 0.
int Npc_GetDistToNpc (c_npc npc1, c_npc npc2); - возвращает расстояние в см. между npc1 и npc2.
int Npc_GetDistToWP (c_npc self, string wpName); - возвращает расстояние в см. между НПС self и WP wpName.
int Npc_GetDistToItem (c_npc self, c_item item); - возвращает расстояние в см. между НПС self и предметом item.
int Npc_GetDistToPlayer (c_npc npc1); - возвращает расстояние в см. между npc1 и ГГ.
void Npc_MemoryEntry (c_npc self, int source, c_npc offender, int newsid, c_npc victim); - данная функция записывает для НПС self новости, где source – источник новостей (определено два источника: NEWS_SOURCE_WITNESS – есть свидетель произошедшего, NEWS_SOURCE_GOSSIP – сплетня), newsid – идентификатор новости (определены новости: NEWS_MURDER - убийство, NEWS_ATTACK – атака, NEWS_THEFT – воровство, NEWS_DEFEAT – поражение, NEWS_NERVE – переживание, NEWS_INTERFERE – вмешательство, NEWS_HASDEFEATED – победа), offender – НПС преступник, victim – НПС жертва.
void Npc_MemoryEntryGuild (c_npc self, int source, c_npc offender, int newsid, c_npc victimguild); - работает аналогично предыдущей функции, только жертвой является вся гильдия.
int Npc_HasNews (c_npc self, int newsID, c_npc offender, c_npc victim); - эта функция проверяет, имеет ли НПС self новость с идентификатором newsID о жертве victim и преступнике offender, возвращает 1, если имеет, иначе – 0. (Вместо ненужных параметров можно записать 0).
int Npc_IsNewsGossip (c_npc self, int newsNumber); - возвращает 1, если новость для НПС self является сплетней, иначе – 0.
c_npc Npc_GetNewsWitness (c_npc self, int newsNumber); - возвращает ссылку на свидетеля в новости newsNumber для НПС self.
c_npc Npc_GetNewsVictim (c_npc self, int newsNumber); - возвращает ссылку на жертву в новости newsNumber для НПС self.
c_npc Npc_GetNewsOffender (c_npc self, int newsNumber); - возвращает ссылку на преступника в новости newsNumber для НПС self.
int Npc_DeleteNews (c_npc self, int newsNumber); - удаляет новость newsNumber для НПС self, возвращает 1, исли удаление успешно, иначе – 0.***
void Npc_ChangeAttribute (c_npc self, int atr, int value); - изменяет значение атрибута atr на кол-во единиц value для НПС self.
void Npc_CreateSpell (c_npc self, int spellnr); - НПС self создает заклинание spellnr, оно становится активным, но еще не применяется.
void Npc_LearnSpell (c_npc self, int spellnr); - НПС self выучил заклинание spellnr и может его использовать.
int Npc_GetActiveSpell (c_npc self); - возвращает номер заклинания, которое имеет активным НПС self, иначе - -1.
int Npc_GetActiveSpellCat (c_npc self); - возвращает номер категории активного заклинания у НПС self. Существуют три категории заклинаний:SPELL_GOOD, SPELL_NEUTRAL, SPELL_BAD.
int Npc_SetActiveSpellInfo (c_npc npc, int i1); - задает любое значение i1 для активного заклинания у НПС npc, это значение может использоваться в скриптах, экзешник на него не реагирует. Возвращаемое значение неизвестно.
int Npc_GetActiveSpellLevel (c_npc self); - возвращает уровень активного заклинания у НПС self.
int Npc_HasSpell (c_npc self, int spellID); - возвращает 1, если НПС self может использовать заклинание spellID, иначе – 0.
void Npc_PercEnable (c_npc self, int percID, func function); - функция активации восприятия percID у НПС self, где function – функция обработки восприятия.
void Npc_PercDisable (c_npc self, int percID); - функция деактивации восприятия percID у НПС self.
void Npc_SetPercTime (c_npc self, float seconds); - установка времени реакции в секундах seconds НПС self на событие для активного восприятия.
void Npc_SendPassivePerc (c_npc npc1, int Perc_type, c_npc npc2, c_npc npc3); - функция посылки пассивного восприятия Perc_type от npc1, где npc2 – жертва, npc3 – преступник.
void Npc_SendSinglePerc (c_npc self, c_npc target, int percID); - функция посылки восприятия percID от НПС self к НПС target.
void Npc_PerceiveAll (c_npc self); - разрешает НПС self воспринимать все объекты в зоне действия восприятия, затем можно использовать функции Wld_DetectNpc и Wld_DetectItem.
string Npc_GetDetectedMob (c_npc self); - возвращает имя MOB (Move Object) объекта, который распознал НПС self. Например: если имя MOB объекта " DOOR_OCR _135", то функция возвратит " DOOR ".
int Npc_CanSeeNpc (c_npc npc1, c_npc npc2); - возвращает TRUE, если npc1 может видеть npc2, иначе – FALSE.
int Npc_CanSeeNpcFreeLOS (c_npc self, c_npc other); - возвращает TRUE, если НПС self может видеть НПС other по прямой, без учета угла обзора, иначе – FALSE.
int Npc_CanSeeItem (c_npc npc1, c_item item); - возвращает TRUE, если npc1 может видеть предмет item, иначе – FALSE.
int Npc_CanSeeSource (c_npc self); - возвращает TRUE, если НПС self может видеть источник звука, иначе – FALSE.
int Npc_IsPlayerInMyRoom (c_npc npc); - возвращает TRUE, если НПС npc находится в своем помещении или в помещении своей гильдии, иначе – FALSE.
int Npc_WasPlayerInMyRoom (c_npc npc); - возвращает TRUE, если НПС npc находился в своем помещении или в помещении своей гильдии перед изменением своего положения, иначе – FALSE.
int Npc_GetComrades (c_npc npc); - возвращает 1, если НПС npc имеет друзей, иначе – 0. ***
int Npc_IsDetectedMobOwnedByNpc (c_npc user, c_npc owner); - возвращает значение > 0, если НПС owner является владельцем MOB а, который использует НПС user.
int Npc_IsDetectedMobOwnedByGuild (c_npc user, int ownerguild); - возвращает значение > 0, если гильдия ownerguild является владельцем MOB а, который использует НПС user.
void Npc_GiveItem (c_npc self, c_item item, c_npc other); - НПС self получает предмет item от НПС other.
int Npc_StartItemReactModules (c_npc self, c_npc other, c_item item); - проверяет все модули ItemReact реакции на предмет item, полученный НПС self от НПС other, запускает соответствующую функцию Reaction и возвращает TRUE, если находит модуль, иначе – FALSE.
int Npc_HasOffered (c_npc self, c_npc other, int itemInstance); - проверяет, имеет ли НПС other предмет itemInstance для передачи НПС self, возвращает TRUE, если предмет имеется, иначе – FALSE.
c_item Npc_GetInvItem (c_npc self, int itemInstance); - получить ссылку на предмет, который имеет НПС self с номером itemInstance.
int Npc_HasItems (c_npc n0, int itemInstance); - возвращает количетво предметов itemInstance у НПС n0.

int Npc_GetInvItemBySlot (c_npc self, int category, int slotNr); - возвращает кол- во предметов, которые находятся у НПС self, где category – категория инвентаря (INV_WEAPON, INV_ARMOR, INV_RUNE, INV_MAGIC, INV_FOOD, INV_POTION, INV_DOC, INV_MISC), slotNr – номер слота предмета.
void Npc_RemoveInvItem (c_npc owner, int itemInstance); - предмет itemInstance удаляется из инвентаря НПС owner и из игры.
void Npc_RemoveInvItems (c_npc owner, int itemInstance, int amount); - указанное кол-во amount предметов itemInstance удаляется из инвентаря НПС owner и из игры.
c_item Npc_GetEquippedMeleeWeapon (c_npc n0); - возвращает оружие ближнего радиуса поражения, которым экипирован НПС n0.
c_item Npc_GetEquippedRangedWeapon (c_npc n0); - возвращает оружие дальнего радиуса поражения, которым экипирован НПС n0.
c_item Npc_GetEquippedArmor (c_npc n0); - возвращает доспехи, которыми экипирован НПС n0.
int Npc_HasEquippedWeapon (c_npc self); - возвращает 1, если НПС self экипирован оружием, иначе - 0.
int Npc_HasEquippedMeleeWeapon (c_npc self); - возвращает 1, если НПС self экипирован оружием ближнего радиуса поражения, иначе - 0.
int Npc_HasEquippedRangedWeapon (c_npc self); - возвращает 1, если НПС self экипирован оружием дальнего радиуса поражения, иначе - 0.
int Npc_HasEquippedArmor (c_npc self); - возвращает 1, если НПС self экипирован доспехами, иначе - 0.
void Npc_SetToFistMode (c_npc self); - ставит НПС self в режим кулачного боя.
void Npc_SetToFightMode (c_npc self, int weapon); - ставит НПС self в режим боя с соответствующим оружием weapon.
int Npc_IsInFightMode (c_npc self, int fmode); - возвращает 1, если НПС self находится в боевом режиме fmode, иначе - 0. Заданы следующие боевые режимы: FMODE_NONE – небоевой режим, FMODE_FIST – режим кулачного боя, FMODE_MELEE – боевой режим с оружием ближнего радиуса поражения, FMODE_FAR - боевой режим с оружием дальнего радиуса поражения, FMODE_MAGIC – боевой режим с магией.
c_item Npc_GetReadiedWeapon (c_npc n0); - возвращает ссылку на оружие, которое НПС n0 держит в руке.
int Npc_HasReadiedWeapon (c_npc self); - возвращает 1, если НПС self держит любое оружие в руке, иначе - 0.
int Npc_HasReadiedMeleeWeapon (c_npc self); - возвращает 1, если НПС self держит в руке оружие ближнего радиуса поражения, иначе - 0.
int Npc_HasReadiedRangedWeapon (c_npc self); - возвращает 1, если НПС self держит в руке оружие дальнего радиуса поражения, иначе - 0.
int Npc_HasRangedWeaponWithAmmo (c_npc npc); - возвращает 1, если НПС npc держит в руке или имеет в инвентаре оружие дальнего радиуса поражения с боеприпасами, иначе - 0.
int Npc_GetTarget (c_npc self); - возвращает TRUE, если НПС self имеет цель для поражения (в качестве цели выступает НПС other), иначе – FALSE.
int Npc_GetNextTarget (c_npc self); - выполняется активный поиск цели для НПС self. Если цель находится, то она становится внутренней целью и записывается в переменную
other, если цель не найдена, то внутренняя цель удаляется и other становится недействительным. Критерий поиска цели - в качестве цели возмется враждебный противник, который не мертв или находится не в бессознательном состоянии. Возвращает TRUE, если цель найдена, иначе – FALSE. Внимание: Поиск основывается на активном восприятии НПС self, поэтому, если активное восприятие не установлено, то сначала следует применять функцию Npc_PerceiveAll ().
int Npc_IsNextTargetAvailable (c_npc self); - работает аналогично предыдущей функции, только ни внутренняя цель, ни переменная other не инициализируются.
void Npc_SetTarget (c_npc self, c_npc other); - устанавливает для НПС self в качестве внутренней цели для поражения НПС other.
int Npc_AreWeStronger (c_npc self, c_npc other); - выявляет более сильного НПС среди self и other по следующему алгоритму: если сумма уровней всех НПС (людей и монстров), которые враждебны к self и дружелюбны к other, более чем в два раза превышает сумму уровней всех НПС (людей и монстров), которые дружелюбны к self и враждебны к other, то other сльнее и возвращается FALSE, иначе сильнее self и возвращается TRUE. Замечания: 1. НПС, которые враждебны к обоим, не учитываются. 2. НПС, который враждебен к одному и дружелюбен к другому, будет участвовать в подсчете два раза.
int Npc_IsAiming (c_npc self, c_npc other); - возвращает 1, если НПС other целится в НПС self из оружия дальнего радиуса поражения или магией, иначе – 0.
int Npc_GetTalentSkill (c_npc self, int talent); - возвращает уровень навыка talent, который имеет НПС self.*** Определены следующие способности: NPC_TALENT _1 H – владение одноручником, NPC_TALENT _2 H – владение двуручником, NPC_TALENT_BOW – владение луком, NPC_TALENT_CROSSBOW – владение арбалетом, NPC_TALENT_PICKLOCK – умение вскрывать замки, NPC_TALENT_PICKPOCKET – карманная кража, NPC_TALENT_MAGE – владение магией, NPC_TALENT_SNEAK – умение подкрадываться, NPC_TALENT_REGENERATE – способность восстанавливать здоровье, NPC_TALENT_FIREMASTER – способность мастерски обращаться с огнем, NPC_TALENT_ACROBAT – акробатика.
int Npc_GetTalentValue (c_npc self, int talent); - возвращает кол-во единиц навыка talent, которые имеет НПС self.***
void Npc_SetTalentSkill (c_npc self, int talent, int level); - установить уровень level навыка talent, которым владеет НПС self.***
void Npc_SetTalentValue (c_npc self, int talent, int value); - установить кол-во единиц value навыка talent, которым владеет НПС self.***
Вот мы и закончили рассмотрение большого раздела – функции работы с НПС.

15. Функции искусственного интеллекта (AI_ функции).

void AI_Wait (c_npc n0, float n1); - НПС n0 переводится на n1 секунд в состояние ожидания, в этом состоянии он не делает ничего, но наносимый ущерб регистрируется и работает пассивное восприятие.
void AI_PlayAni (c_npc n0, string s0); - для НПС n0 проигрывается файл анимации s0 (имя файла s0 обязательно пишется заглавными буквами).
void AI_StandUp (c_npc self); - если НПС self находится в состоянии анимации, то проигрывается соответствующее движение, если НПС self находится на MOBSI объекте (стул, кровать и т.д.), то он встает.
void AI_StandUpQuick (c_npc self); - в отличии от предыдущей функции никаких движений не проигрывается, НПС self сразу переводится в стоячее состояние.
void AI_QuickLook (c_npc self, c_npc other); - НПС self быстро бросает взгляд на НПС other. Выполняется только движение головой в течение 2 секунд.
void AI_LookAt (c_npc self, string name); - НПС self смотрит на определенную точку на местности или на объект, где name – имя точки местности или имя VOB объекта.
void AI_LookAtNpc (c_npc self, c_npc other); - НПС self смотрит на НПС other.
void AI_StopLookAt (c_npc self); - НПС self снова смотрит прямо перед собой.
void AI_PointAt (c_npc self, string name); - НПС self указывает на определенную точку на местности или на объект, где name – имя точки местности или имя VOB объекта.
void AI_PointAtNpc (c_npc self, c_npc other); - НПС self указывает на НПС other.
void AI_StopPointAt (c_npc self); - НПС self прекращает указывать на что-нибудь или кого-нибудь.
void AI_TakeItem (c_npc self, c_item item); - НПС self берет (принимает) предмет item.
void AI_DropItem (c_npc self, int itemid); - НПС self выбрасывает на землю или на пол предмет itemid.
void AI_UseItem (c_npc self, int itemInstance); - НПС self использует предмет itemInstance.
void AI_UseItemToState (c_npc self, int itemInstance, int state); - НПС self использует предмет itemInstance до указанного состояния state (-1 – используется полностью).
void AI_TakeMob (c_npc self, string name); - НПС self берет (принимает) MOB (рюкзак, корзину и т.д.) с именем name.***
void AI_DropMob (c_npc self); - НПС self выбрасывает на землю или на пол имеющийся MOB.***
int AI_UseMob (c_npc self, string schemeName, int targetState); - НПС self использует MOB с именем schemeName до указанного состояния targetState. Если указанное состояние у MOBa уже имеется, НПС self приближается к нему, но ничего не делает.
void AI_Waitms (c_npc self, int msec); - НПС self переводится на msec миллисекунд в состояние ожидания, в этом состоянии он не делает ничего, но наносимый ущерб регистрируется и работает пассивное восприятие.***
void AI_CanSeeNpc (c_npc self, c_npc other, func see); - НПС self может увидеть НПС other, где see – какая-то функция (описания не нашел).***
void AI_SetWalkmode (c_npc n, int n0); - установить режим передвижения n0 для НПС n. Определены следующие режимы: NPC_RUN – бег, NPC_WALK – ходьба шагом, NPC_SNEAK – подкрадывание, NPC_RUN_WEAPON – бег с оружием, NPC_WALK_WEAPON – ходьба с оружием, NPC_SNEAK_WEAPON – подкрадывание с оружием.
void AI_GotoWP (c_npc n0, string s0); - НПС n0 перемещается в указанную WP s0.
void AI_GotoFP (c_npc self, string fpName); - НПС n0 перемещается в указанную FP fpName, расположенную в пределах 20 метров. Критерий поиска FP аналогичен используемому в функции Wld_IsFPAvailable.
void AI_GotoNextFP (c_npc self, string fpName); - работает аналогично предыдущей функции, только критерий поиска FP аналогичен используемому в функции Wld_IsNextFPAvailable.
void AI_GotoNpc (c_npc self, c_npc other); - НПС self перемещается к НПС other.
void AI_GotoItem (c_npc self, c_item item); - НПС self перемещается к предмету item.
void AI_GotoSound (c_npc n0); - НПС self перемещается к источнику звука.
void AI_Teleport (c_npc self, string waypoint); - телепортирует НПС self в новую локацию waypoint.

void AI_TurnToNpc (c_npc n0, c_npc n1); - НПС n0 поворачивается к НПС n1 лицом.
void AI_TurnAway (c_npc n0, c_npc n1); - НПС n0 поворачивается к НПС n1 спиной.
void AI_WhirlAround (c_npc self, c_npc other); - НПС self быстро поворачивается к НПС other лицом. (Делает оборот).
void AI_WhirlAroundToSource (c_npc self); - НПС self быстро поворачивается лицом к какому-то источнику.*** (Описания не нашел).
void AI_TurnToSound (c_npc self); - НПС self поворачивается лицом к источнику звука.
void AI_AlignToWP (c_npc self); - НПС self выравнивается в WP точке по направлению стрелки, заданной в Spacer.
void AI_AlignToFP (c_npc self); - НПС self выравнивается в FP точке по направлению стрелки, заданной в Spacer.***
void AI_Dodge (c_npc npc); - НПС npc отклоняется назад. (Способ защиты при атаке противника).
void AI_PlayAniBS (c_npc npc, string aniname, int bodystate); - для НПС npc проигрывается файл анимации aniname для определенного состояния тела bodystate.
void AI_PlayCutscene (c_npc self, string csName); - для НПС self проигрывается Cutscene с именем csName, заданная в скриптах.
void AI_DrawWeapon (c_npc n0); - НПС n0 вытаскивает оружие, которым экипирован.
void AI_RemoveWeapon (c_npc n0); - НПС n0 прячет оружие.
void AI_ReadyMeleeWeapon (c_npc self); - НПС self готовит оружие ближнего радиуса поражения к бою.
void AI_ReadyRangedWeapon (c_npc self); - НПС self готовит оружие дальнего радиуса поражения к бою.
void AI_Attack (c_npc self); - НПС self начинает сражение (эта функция должна вызываться внутри ZS_Attack_Loop). Атакуется внутренняя цель, которая была задана функцией Npc_SetTarget или Npc_GetNextTarget.
void AI_FinishingMove (c_npc self, c_npc other); - логическое завершение операции приближения (поворота) НПС self к НПС other.
void AI_Defend (c_npc self); - НПС self парирует удар противника (защищается). Выполняется только во время атаки противника.
void AI_Flee (c_npc self); - НПС self убегает от противника (эта функция должна вызываться внутри ZS_Loop). Предварительно функцией Npc_SetTarget должна быть установлена внутренняя цель, от которой НПС self должен убежать.
void AI_AimAt (c_npc attacker, c_npc target); - НПС attacker целится из оружия дальнего радиуса поражения в НПС target.
void AI_StopAim (c_npc attacker); - НПС attacker прекращает целиться из оружия дальнего радиуса поражения.
void AI_ShootAt (c_npc attacker, c_npc target); - НПС attacker стреляет из оружия дальнего радиуса поражения в НПС target.
void AI_CombatReactToDamage (c_npc self); - реакция НПС self на повреждение во время боя.***
void AI_LookForItem (c_npc self, int instance); - функция показывает определенный предмет instance, который имеется у НПС self. (Например: золотой меч разрушения).
void AI_EquipBestMeleeWeapon (c_npc self); - в инвентаре НПС self ищется самое лучшее оружие ближнего радиуса поражения и вешается на пояс.
void AI_EquipBestRangedWeapon (c_npc self); - в инвентаре НПС self ищется самое лучшее оружие дальнего радиуса поражения и вешается на спину.
void AI_EquipBestArmor (c_npc self); - в инвентаре НПС self ищутся и одеваются самые лучшие доспехи.
void AI_UnequipWeapons (c_npc self); - все экипированное оружие НПС self убирается в инвентарь.
void AI_UnequipArmor (c_npc self); - одетые доспехи НПС self убираются в инвентарь.
void AI_EquipArmor (c_npc self, int armor); - НПС self одевает доспехи armor, которые должны находиться в его инвентаре.***
void AI_ReadySpell (c_npc self, int spellID, int investMana); - НПС self берет в руку заклинание spellID со стоимостью маны investMana.
void AI_UnreadySpell (c_npc self); - НПС self прячет в инвентарь заклинание, которое имеет в руке.
void AI_Output (c_npc self, c_npc target, string outputName); - данная функция реализует диалог, НПС self говорит фразу outputName для НПС target. Текст фразы должен располагаться в скриптах в виде комментария к функции. (например: AI_Output () //текст фразы).
void AI_OutputSVM (c_npc self, c_npc target, string svmname); - данная функция реализует SVM (Standart Voice Module) диалог, НПС self говорит фразу svmname для НПС target. Текст фразы должен располагаться в скриптах в файле svm.d (в одном из соответствующих SVM_ модулей) в виде комментария к строке с именем svmname. (например: StopMagic = svmname; //текст фразы).
void AI_OutputSVM_Overlay (c_npc self, c_npc target, string svmname); -выполняется аналогично предыдущей функции, фраза выдается быстро, нет ожидания при разговоре следующей AI_ команды. Используется для комментариев перед сражением и во время сражения.
void AI_WaitTillEnd (c_npc self, c_npc other); - НПС self ждет от НПС other ответа на свою фразу. (Не выполняется перед функцией AI_OutputSVM_Overlay).
void AI_Ask (c_npc self, func answerYes, func answerNo); - определяет, как НПС self будет отвечать на сказанную фразу, при выборе ответа «Да» - выполняется функция answerYes, иначе – функция answerNo. Эти функции должны быть описаны заранее.
void AI_AskText (c_npc self, func funcYes, func funcNo, string strYes, string strNo); - работает аналогично предыдущей функции, только для вариантов ответов (кроме функций) могут быть заданы фразы ответов strYes, strNo.
void AI_WaitForQuestion (c_npc self, func scriptFunc); - НПС self ждет ответа или вопроса в течение 20 секунд, если в это время он поступает, то выполняется функция scriptFunc.
void AI_ProcessInfos (c_npc self); - эта функция запускает диалог для НПС self с возможностью выбора вариантов ответов/вопросов.***
void AI_StopProcessInfos (c_npc npc); - режим диалога НПС npc, начатый предыдущей функцией, заканчивается.
void AI_StartState (c_npc self, func what, int stateBehaviour, string wpName); - переводит НПС self в соответствующее состояние, описанное функцией what, где stateBehaviour = 0 – выполняется последовательность состояний, 1 – выполняется только функция конечного состояния, wpName – имя WP точки (если "", то состояние изменяется на месте расположения НПС).
void AI_SetNpcsToState (c_npc self, func aiStateFunc, int radius); - переводит всех НПС, находящихся от НПС self на расстоянии radius сантиметров, в соответствующее состояние, описанное функцией aiStateFunc.
void AI_ContinueRoutine (c_npc self); - продолжает выполнение распорядка дня НПС self.

Тема седьмая: Встроенные функции Готики 2.

В основном в Готике 2 используются функции Готики 1, но есть несколько дополнительных функций, их мы и рассмотрим.
int AI_PrintScreen (string msg, int posx, int posy, string font, int timeSec); - выводит на экран строку текста msg (имя шрифта – font) с координатами posx, posy (диапазон от 0 до 99% размера экрана, -1 означает вывод по центру соответствующей оси экрана) на время timeSec (в секундах).
void AI_Snd_Play (c_npc self, string s0); - воспроизвести звуковой файл с именем s0 для НПС self.
void AI_Snd_Play3D (c_npc self, c_npc n0, string s0); - воспроизвести звуковой файл с именем s0 для НПС self.
void AI_PlayFX (c_npc self, c_npc n0, string s0); - воспроизвести видео-файл с именем s0 для НПС self.
void AI_StopFX (c_npc self, string s0); - остановить воспроизведение видео-файла с именем s0 для НПС self.
int PlayVideoEx (string video, int aa, int bb); - показать видео файл video (с расширением BIK), возвращает 1, если видео показано, 0 – в случае ошибки. Трактование параметров aa и bb не знаю, в скриптах они имеют значение TRUE и FALSE.
void ExitSession () – используется вместо функции ExitGame (), хотя она тоже сохранена.
int GameInitGerman () – просто всегда возвращает 1.
В английской версии экзешника еще есть функции - int GameInitEnglish () и int GameInitEngintl (), их назначение мне неизвестно.
int Wld_DetectNpcExAtt (c_npc self, int instance, func aiState, int guild, int detectPlayer, int xxx); - эта функция инициализирует глобальную переменную скриптов other, отличную от НПС self, где instance – производная от класса c_npc, которая должна быть найдена и проинициализирована (-1 – любая производная), guild – гильдия, членом которой должен быть искомый НПС (-1 – любая гильдия), aiState – функция AI состояния, в котором должен находится искомый НПС (NOFUNC – любое AI состояние). Функция возвращает 1 в случае успешного завершения (other инициализирован найденным НПС), 0 – неудача (other не определен), параметр detectPlayer указывает, исключить ли игрока (ГГ) из поиска (0 – исключить, 1 – нет). В этой функции, по сравнению с Wld_DetectNpcEx, добавлен параметр xxx, к сожалению мне про него ничего не известно (возможно это какой-нибудь атрибут).
int Wld_IsRaining (); – возвращает 1, если идет дождь, иначе – 0.
void Wld_StopEffect (string name); - останавливает проигрывание эффекта с именем name, запущенного функцией Wld_PlayEffect.
void Doc_SetLevelCoords (int Document, int Left, int Top, int Right, int Bottom); - устанавливает границы области, отображаемой на карте (границы задаются в абсолютных координатах уровня), где: Document – обрабатываемый документ, Left, Top, Right и Bottom – задаваемые границы.
int Npc_GetLastHitSpellID (c_npc self); - возвращает номер заклинания, которое НПС self применил последним.
int Npc_GetLastHitSpellCat (c_npc self); - возвращает категорию заклинания, которое НПС self применил последним.
int Npc_GetActiveSpellIsScroll (c_npc self); - проверяет, является ли активное заклинание у НПС self свитком, если да – возвращает 1, иначе – 0.
void Npc_ClearInventory (c_npc self); - очищает весь инвентарь у НПС self.
int Npc_GetHeightToNpc (c_npc npc1, c_npc npc2); - возвращает расстояние в см. по высоте между НПС npc1 и npc2.
int Npc_GetHeightToItem (c_npc self, c_item item); - возвращает расстояние в см. по высоте между НПС self и предметом item.
c_npc Npc_GetLookAtTarget (c_npc self); - инициализирует глобальную переменную target ссылкой на НПС, на которого смотрит НПС self.

void Npc_StopAni (c_npc self, string name); - остановить проигрывание файла анимации с именем name для НПС self.
int Npc_IsDrawingWeapon (c_npc self); - возвращает номер оружия, которое НПС self держит в руках, иначе – 0.
int Npc_IsDrawingSpell (c_npc self); - возвращает номер заклинания, которое применяет НПС self, иначе – 0.
int Npc_IsInPlayersRoom (c_npc self); - возвращает 1, если НПС self находится в помещении, которое принадлежит ГГ, иначе – 0.
c_npc Npc_GetPortalOwner (c_npc self); - возвращает НПС-владельца помещения (области), в котором находится НПС self.
int Npc_GetPortal Guild (c_npc self); - возвращает номер гильдии владельца помещения (области), в котором находится НПС self.