LexiconJP

Contributors

このリストに名前が挙げられている人物達は、Lexiconを作るために少なからず彼らのプライベートな時間を費やしました。ファンクションの働きを明らかにするために、彼らは長い時間をかけて調査しました。我々が望むことは、より良いMOD製作の為にこのLexiconが何らかの役に立つことです。

LexiconJP team:

akito
geshi
katsu
marimo
marshall
Masacy
NamaYake
ngtaicho
Rainie
Redondo
ryuga
Tizumi
Volac
weed

Lexicon International:

Alex Skinner
Bill Castello
Brett Lathrope
Celowin Content
Charles Feduke
Chaz Mead
Craig Smith
Dan Spezzano
Daniel Beckman
Dave Robinson
Dave Withnall
David Downing
David Gaider
Drake Coker
GoLeM
Graziano Lenzi
Herner Klenthur
Iskander Merriman
Jason Harris
Jason Simpson
Jeff Lindsey
Jeremy Ludwig
Jeremy Spilinek
Jochem van 't Hull
Jody Fletcher
John Shuell
Kenneth Toomey
Kevin Rogala
Kristian Markon
Lilac Soul
Maximus
Michael Kenyon
Michael Nork
Neil Burton
Paul Thomas
Peter
Peter Westergaard
Richard Conner
Roger Sandbakk
Ryan Hunt
Sarev0k
Steve Moseley
Timothy Tryzbiak
Tom Cassiotis
Troels Therkelsen

License
Copyright (c) 2002-2003 NWN Lexicon Group. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation. A copy of the license is included in the section entitled "License".

Neverwinter Nights (c) 2002 Infogrames Entertainment, S.A. Some code examples c 2001-2002 BioWare Corp. BioWare, BioWare Aurora Engine are trademarks of BioWare Corp. Neverwinter Nights, Forgotten Realms, Dungeons & Dragons, and D&D are trademarks owned by Wizards of the Coast, Inc., a subsidiary of Hasbro, Inc. All other trademarks are the property of their respective owners.

NWN C-Script Function Categories
ファンクション　カテゴリー別

変数
01.Local Variables (28)
02.Math (22)
03.String (14)
04.Type Casting/Conversion (13)

関数
05.Function (49)
06.Private Functions (11)

オブジェクト
07.Action on Object (76)
08.Targeting (19)

データ取得
09.Get Data (32)
10.Get Data from Creature (94)
11.Get Data from Object (48)

プレーヤーキャラクター関連
12.Alignment (11)
13.Conversation (23)
14.Experience (9)
15.Journal (3)
16.Money (10)
17.PC Only (23)

アイテム
18.Inventory (46)
19.Item Creation (99)

キャラクター
20.Henchmen/Familiars/Summoned (63)
21.Party (21)
22.Reputation/Faction (41)

行動・技能
23.Movement (34)
24.Perception (14)
25.Saving Throw (10)
26.Spells (47)
27.Talents/Skills/Feats (34)

AI
28.Core AI (67)
29.Core AI Talent (20)

戦闘
30.Combat (19)
31.Combat Actions (19)
32.Combat Information (25)

効果
33.Cut-Scene (5)
34.Effects (107)
35.Lighting Effects (8)
36.Music Effects (13)
37.Sound Effects (15)
38.Spell Casting Effects (24)
39.Visual Effects (15)

場所に関係するもの
40.Area (29)
41.Encounter (12)
42.Module (24)
43.Module Specific (82)
44.Traps (18)

その他
45.Database (3)
46.Debug (17)
47.Miscellaneous (43)
48.Prototyped but Unused (17)
49.Server (6)
50.Time (19)

Objects with Events

イベントのあるオブジェクトはNWNで使われるオブジェクトの大半を占めます。

NWNのプログラマー達はオブジェクトにイベントを関連付けることによって、PC達と他のオブジェクトとの相互作用を実現できるようにしました。その方法は個々のスクリプトをそれぞれのイベントに割り当てます。

Objects

Area (エリア)

エリアはMOD内でのタイルセットを基本とした一単位です。エリア内にはイベントに関連したオブジェクトが多数配置されています。
エリアイベントではOBJECT_SELFの意味するものはそのイベントが起こったエリア自身です。
通常MODは複数のエリアで構成されています。

Creature (クリーチャー)

コメントを削除することによって使用できます（もしくはOnUserDefinedにスクリプトを書き込みます）。
これによってクリーチャーの行動が予想通りになり、より複雑な行動をさせることもできるようになります。

Door (ドア)

ドアに対するイベント

Encounter (エンカウンター)

エンカウンターは自動的にクリーチャーを出現させるトリガーです。
ここではエンカウンターの修正やPCのスキル・レベルに応じてランダムに出現させるようにできます。

Module (モジュール)

モジュールは技術的であり、冒険そのものであり、複数のMODをつなげてキャンペーンとすることもできます。
NWNの常設世界でもキャンペーンができますし、またあるサーバーで異なった時間に一つの巨大なMODや複数のMODを連続させることもできます。
OBJECT_SELFはイベントの起こったエリアをあらわします（MOD自身そのものではありません）。

Placeable Object (配置用オブジェクト)

配置用オブジェクトはNWNのMODに深みを与え、スクリプトを割り当てることによって様々なイベントを起こすことができます。
ここにはアイテムが含まれ、トリガーのように扱われたり、鍵や罠を仕掛けることもできます。
OBJECT_SELFはイベントを起こす配置用オブジェクト自身で、イベントを惹起したクリーチャーや他のオブジェクトではありません。
配置用オブジェクトはある行動を起こすこともできます（例えばActionPlayAnimation()がANIMATION_PLACEABLE_*定数を参照する場合など）。しかし本質的に不可能な行動を指定しようとすると（つまりActionLockObject(), ActionUnlockObject(), ActionEquipItem(), ActionPickUpItem()のこと）問題を引き起こします。
全ての配置用オブジェクトが空けたり/閉じたり、稼動（アクティベート）/停止（ディアクティベート）させられるとは限りません。

Trap (トラップ)

トラップ（罠）は解除可能、発見可能な装置です。発見も解除もされずにPCが引っかかると結構なダメージを与えていやな思いをさせられます。その上ダメージだけでなく追加効果もあるかもしれません。

Trigger (トリガー)

トリガーは三種類あります：エリア・トランジション、一般トリガー、トラップ。
エリア・トランジションはキャラクターを別の場所へ移動させます。
一般トリガーは特殊なイベントを起こすためにスクリプトを割り当てることができます。
トラップは床に指定し、そこに踏み込んだ者に何か不愉快な思いをさせることができます。
OBJECT_SELFはトリガー自身であって、トリガーを引き起こしたオブジェクトではありません。

Constants

定数とは数値を表す名称です。

定数は特定の値を表す記述として一ヶ所で定義しておけば、ソースコード上のその定数は全て同じ数値として振る舞います。
例えば年数を表す数値をソースコード上で扱います。その値を2002から2003に変更しようとします。
ソースコード上のあちこちにある2002という記述を2003に変更するよりも、定数"YEAR"をソースコード上に用いておけば、あとから"YEAR"の定義を2002から2003に変更するだけで済むのです。

また、定数は一意の名前を付けることにより、人間が解釈しやすいようにするといった用途でも用いられます。
例えば、白を表す数値(0xFFFFFF)の代わりにWHITEと書いた方が解釈しやすいでしょう。

定数は変数と異なり一定の値をとり、その値を動的に変えることはできません。

コンパイル時にプリコンパイラーがソースコード上の定数を意味するコードを探し出して実際の数値に置き換えます。

Constant Group

ABILITY_*
AC_*
ACTION_*
ALIGNMENT_*
ANIMAL_COMPANION_CREATURE_TYPE_*
ANIMATION_*
AOE_MOB_*
AOE_PER_*
AREA_TRANSITION_*
ASSOCIATE_COMMAND_*
ASSOCIATE_TYPE_*
ATTACK_BONUS_*
ATTITUDE_*
BASE_ITEM_*
BODY_NODE_*
CAMERA_MODE_*
CAMERA_TRANSITION_TYPE_*
CLASS_TYPE_*
COMBAT_MODE_*
CREATURE_SIZE_*
CREATURE_TYPE_*
DAMAGE_BONUS_*
DAMAGE_POWER_*
DAMAGE_TYPE_*
DIRECTION_*
DISEASE_*
DOOR_ACTION_*
DURATION_TYPE_*
EFFECT_TYPE_*
ENCOUNTER_DIFFICULTY_*
FAMILIAR_CREATURE_TYPE_*
FEAT_*
GAME_DIFFICULTY_*
GENDER_*
Globals
GUI_PANEL_*
IMMUNITY_TYPE_*
INVENTORY_DISTURB_TYPE_*
INVENTORY_SLOT_*
INVISIBILITY_TYPE_*
ITEM_PROPERTY_*
METAMAGIC_*
NW_ASC_*
NW_ASC_AGRESSIVE_*
NW_ASC_DISTANCE_*
NW_ASC_HEAL_AT_*
NW_FLAG_*
NW_FLAG_BEHAVIOR_*
OBJECT_TYPE_*
PANEL_BUTTON_*
PERCEPTION_*
PERSISTENT_ZONE_*
PLACEABLE_ACTION_*
PLAYER_CHAR_*
POISON_*
POLYMORPH_TYPE_*
PROJECTILE_PATH_TYPE_*
RACIAL_TYPE_*
RADIUS_SIZE_*
REPUTATION_TYPE_*
REST_EVENTTYPE_*
SAVING_THROW_TYPE_*
SHAPE_*
SKILL_*
SPECIAL_ATTACK_*
SPELL_*
SPELL_SCHOOL_*
SPELLABILITY_*
STANDARD_FACTION_*
SUBSKILL_*
SUBTYPE_*
TALENT_CATEGORY_*
TALENT_TYPE_*
TALKVOLUME_*
TILE_MAIN_LIGHT_COLOR_*
TRAP_BASE_TYPE_*
TREASURE_*
VFX_BEAM_*
VFX_COM_*
VFX_DUR_*
VFX_FNF_*
VFX_IMP_*
VOICE_CHAT_*
WEATHER_*

Data Types

データ型とは、値を格納するために使用される異なる変数の型です。
string型は文章だけを格納しますが、int型は小数点を持たない実数を格納します。
異なるデータ型は、異なった多様な使われ方をします。

Data Types

action

An action.

command

A void-returning function, whether it is user defined or engine-defined. This is used by the DelayCommand() and AssignCommand() functions.

constant

定数(constant)とは値を変えることが出来ない数値で、通常は整数値です。
コードの中で直接数値を書くよりも解釈しやすくするために使われる"マジックナンバー"です。

effect

effect型はobjectに対する効果(エフェクト)を表す型です。(視覚的な効果と実際の能力に対する効果のどちらも含みます。)
スクリプト言語内で場所またはobjectに適用したいエフェクトを取得し、
ApplyEffectAtLocation関数、ApplyEffectToObject関数で適用します。

event

event型とはobjectに起こった出来事(イベント)を表すデータ型です。(例：(クリーチャーなどが)攻撃された、話しかけられた、(アイテムなどが)使用された、(ドアなどが)開けられた、etc')
スクリプト実行中にあるobjectにイベントが発生すると、そのobjectに用意されたイベント処理スクリプトが実行されます。
スクリプト記述の中でイベント発行関数(SignalEvent)を用いることで任意のobjectにイベントを発生させることが出来ます。

float

float型は32bit浮動小数値です。
-3.402823e38～3.402823e38の範囲をとり、デフォルトは0.0です。
float型として値の初期化や設定をしたい場合には、必ず小数点を用いて少なくとも小数第一位まで書いてください。
float値を表す小文字のfはあっても無くてもコンパイラは認識します。(例：1.23でも1.23fでも可)

int

int型は符号つき32bit整数値です。
小数値を持たない実数で、マイナス(負)の値を持つことができます。
-2,147,483,648～2,147,483,647の範囲をとり、デフォルトは0です。

location

location型は以下の3つから構成される位置情報です。
・エリアのIDとなるobject型
・エリア内での位置を示すvector型
・向いている方向を示すfloat型

object

ゲーム内にある特定の物体を示す整数値です。
基本的に実体のあるものを指すものです。(クリーチャー、アイテム、ドア、設置可能オブジェクト、etc')

string

string型は不定長の文字列で、NULL文字で終端を表します。(好きな長さの文字を表わせる型で、必ず""(ダブルクウォーテーション)で括る、という意味です)
デフォルト値は空の文字列("")です。
+(プラス記号)を用いることで文字列同士を連結することが出来ます。

struct

異なる変数をまとめ、1つの変数にした構造体。
関数で用いることによって多くの値を一度に返したい場合などに複雑さを軽減します。

talent

talent型は特技、技能または呪文を表すデータ型です。

vector

3つのfloat値からなる構造体で、通常は位置や方向を示すベクトルとして定義されています。
vector型の構成要素を参照するためには、.(ピリオド)演算子を用います。
vector型のデフォルト値は、{ 0.0, 0.0, 0.0 } (x = 0.0、y = 0.0、z = 0.0)です。

void

空の値を示すデータタイプです。
void型を返す関数は実際の値を何も返しません。
アクションキューに登録できる関数はvoid型の関数のみです。