

InformATE!

Manual de Referencia

Zak McKracken
spinf_2000@yahoo.com

Morgul
jmfo1982@yahoo.es

22 de julio de 2003

Resumen

InformATE! (Inform Ahora Totalmente en Español) es una librería para el lenguaje de programación Inform que equivale a la librería original de Graham Nelson, con la particularidad de que (casi) todos los nombres de identificadores están en español y que incluye como parte de la librería estándar algunas características que antes sólo estaban disponibles en módulos separados (acción *Salidas*, admisión de verbos en infinitivo...) y alguna característica nueva, que no existía hasta ahora en ninguna otra parte (como el poder especificar adjetivos para los objetos).

Este es un manual de referencia que lista todos los identificadores y sus equivalencias con los originales en inglés, además de dar información bastante detallada de los conceptos tras cada identificador.

Esto no es un curso de InformATE!. El lector principiante puede recurrir a otros cursillos¹, e incluso a aquellos escritos para la librería en inglés original y usar este manual a modo de diccionario para encontrar las equivalencias con la librería española. Los índices de las últimas páginas serán sin duda muy útiles para este propósito.

¹ Un buen cursillo para principiantes es el de "La Torre: Construye tu propia aventura con InformATE!", escrito por Zak McKracken.

Índice general

1. Introducción	5
1.1. Sobre este documento	5
1.2. Versión de InformATE! documentada	6
1.3. Convenciones tipográficas	7
2. Nombres de fichero	9
3. Atributos	11
4. Propiedades	17
5. Acciones	37
5.1. Meta-acciones	39
5.2. Acciones que no hacen nada	41
5.3. Acciones que pueden llegar a hacer algo	48
5.4. Acciones de depuración	62
5.5. Acciones falsas	63
6. Variables	67
6.1. Variables que el programador debe conocer	67
6.2. Variables internas de la librería	70
7. Objetos	77
8. Rutinas	79
8.1. Rutinas de formato del print	79
8.2. Rutinas que forman parte del lenguaje	80
8.3. Rutinas proporcionadas por la librería	81
8.4. Rutinas que deben programarse en el juego	87
8.5. Rutinas usadas internamente por la librería	90
9. Constantes	95
9.1. Constantes definibles por el juego	95
9.2. Constantes que forman parte del lenguaje	98
9.3. Constantes definidas por la librería	98
9.4. Bits de formato de listas	99
9.5. Códigos de error del parser	100

9.6. Definiciones del idioma	103
9.7. Constantes que representan razones	108
9.8. Otras constantes oscuras	108
10. Índice de identificadores	113
10.1. Identificadores de InformATE!	113
10.2. Equivalencias Inglés - Español	122
10.3. Equivalencias Español - Inglés	130
10.4. Verbos de la gramática	139

Capítulo 1

Introducción

La librería InformATE! (Inform Ahora Totalmente en Español) es una traducción total de la librería original Inform 6/10. Esto significa que no sólo están traducidos los mensajes y la gramática (en los ficheros `Mensajes.h` y `Gramatica.h` respectivamente), sino que también están traducidos todos los nombres de propiedades, atributos, acciones, rutinas, constantes y variables. Incluso algunos de los comentarios están traducidos.

Esta librería además incorpora las extensiones `Salidas.h` e `Infini.h` como parte de la propia librería (no es necesario incluir estos ficheros para disponer de la acción *Salidas* o la comprensión de infinitivos). Además, se ha modificado ligeramente el parser para admitir nuevas propiedades en los objetos que hacen más sencillo y correcto el manejo de su género y número, por lo que ya no es necesario usar la extensión `MultiNom.h` (cuyo propósito era similar).

Además, esta librería permite especificar separadamente el nombre de los objetos y los adjetivos que modifiquen ese nombre, lo cual evita ambigüedades en ciertas situaciones (véase la explicación de la propiedad *adjetivos*).

En definitiva, se trata de una librería diseñada en español y para el programador español (o hispanohablante) que, sin duda, facilitará el aprendizaje de Inform en España e Hispanoamérica.

Como nota importante, señalar que no se permite el uso de acentos ni ñe como parte de un identificador en Inform, por tanto ninguno de los atributos, propiedades, variables, etc. listados en este documento llevan acentos ni ñe.

1.1. Sobre este documento

Este documento es un manual de referencia. No es un cursillo. Por tanto no está escrito para ser leído linealmente ni por personas neófitas, sino como un método de consulta para el programador experimentado, para aquel usuario avanzado que quiera conocer los entresijos del parser o para quien está convirtiendo a InformATE! un juego escrito para la librería inglesa.

Al final del documento viene un diccionario de equivalencias inglés-español y español-inglés de los identificadores (que puede ayudar mucho a los que quieran traducir o portar un juego de la librería inglesa), además de un índice general analítico de todos los identificadores de InformATE! que refiere a las páginas donde se explica cada uno de ellos (los números de página resaltados en negrita) y a las páginas en las que se citan. También se incluye un índice con todos los verbos y formas gramaticales que la librería comprende por defecto.

La mayoría de los identificadores documentados en el manual original de Graham Nelson¹ figuran aquí con una explicación más o menos extensa. En particular, todos los atributos y propiedades aparecen explicados en detalle. Además la sección destinada a las acciones explica con detenimiento los pasos seguidos por la librería para cada verbo del juego.

La sección destinada a rutinas es, de momento, bastante más sucinta y probablemente el lector deberá consultar el manual original en inglés o el manual en español (el "DocumentATE!") para conocer más detalles.

Otras variables y rutinas que sólo usa la librería internamente son listadas para que el lector interesado tenga las equivalencias a mano aunque, de momento, la mayoría están sin explicación de ningún tipo.

1.2. Versión de InformATE! documentada

La versión de la librería a la que se hace referencia en el presente documento es la última en el momento de escribirse, la de número de serie [021230] (30 de Diciembre de 2002). Esta librería se puede obtener de varios sitios en Internet, entre los que están, por orden de accesibilidad y de periodicidad en la actualización:

- La página web del CAAD, dentro de la sección *Descargas* en el apartado *Parsers*:
<http://caad.mine.nu/php/ficheros.php>
O bien desde su sitio FTP:
<ftp://caad.mine.nu/PARSERS>
- La sección *Archivos* del grupo de correo *InformatE* de *Yahoo! Grupos*, que sólo está accesible para los usuarios suscritos a dicho grupo:
<http://es.yahoogroups.com/group/informate/files>
- La página web de Morgul:
<http://www.geocities.com/jmfo1982>
- La página web oficial de InformATE!, mantenida por Zak McKracken, que posiblemente estará desactualizada y no tendrá ninguna versión reciente:
http://www.geocities.com/spinf_2000

¹ Se refiere a "The Inform Designer's Manual".

1.3. Convenciones tipográficas

En este manual de referencia se utilizarán las siguientes tipografías para diferenciar los textos con significado especial y facilitar la lectura del mismo:

Tipo de texto	Muestras de tipografía
Acción	<i>Abrir, DejarSalir</i>
Atributo	nombreplural, soporte
Comando del jugador	«ABRE LA PUERTA», «PÍDELES UN LÁPIZ»
Constante	NO_LUGARES, true
Fragmento de código	<code>print (un) obj, rtrue</code>
Mensaje del juego	“Ya estaba abierto”, “Está muy oscuro”
Nombre de fichero	<code>infixe.h, Msg1P.h</code>
Objeto	brujula, MensajesLibreria
Propiedad	<i>descripcion, esta_en</i>
Rutina	MensajeMuerte, MoverJugador
Texto normal	En este manual de referencia se...
Texto resaltado	Simplemente define algunas <i>acciones falsas...</i>
Título de documento	"The Inform Designer's Manual"
Variable	<i>localizacion_real, modomirar</i>

Capítulo 2

Nombres de fichero

En este capítulo vamos a nombrar y comentar brevemente (sin ningún orden concreto) los ficheros más importantes que forman parte de la librería InformATE!:

EParser.h (Antes *Parser.h*) Es el fichero a incluir al principio del código fuente. Simplemente define algunas *acciones falsas* (en inglés *fake actions*) y unas pocas constantes con información de la versión de la librería. Como ya no se soporta la compilación por módulos, dentro de él se encuentran los antiguos ficheros `enlazlp.h` y `Eparserm.h` (este último es quien realmente contiene el meollo del parser).

Eparserm.h (Antes *parserm.h*) Este fichero contiene el parser de la librería y gran cantidad de funciones de apoyo. Es un fichero muy grande y ampliamente comentado, pero no es necesario examinarlo (¡y mucho menos comprenderlo!) para poder usarlo. Como no se soporta la compilación por módulos, este fichero se encuentra ahora dentro de `EParser.h`.

Acciones.h (Antes *Verblib.h*) Este fichero simplemente define algunas constantes relacionadas con el juego. Dentro de él se encuentra el antiguo fichero `accionm.h`, que es quien contiene el meollo de las acciones.

accionm.h (Antes *verblibm.h*) Este fichero contiene las rutinas que se ejecutarán para cualquier posible acción que el usuario pueda requerir. Cuando el jugador escriba «HUELE FLOR», se ejecutará por ejemplo la rutina `OlerSub`, que se halla definida en este fichero (a menos que el *objeto flor* decida impedirlo capturando la acción *Oler* y reaccionando de forma distinta). Este fichero se encuentra ahora dentro de `Acciones.h`.

Espanol.h (Antes *Spanish.h*) Este fichero contiene funciones específicas del idioma español (como las que escriben los artículos delante de las palabras, teniendo cuidado de que concuerden en género y número) y define las propiedades, atributos, artículos y demás propios de la librería española.

Gramatica.h (Antes *SpanishG.h*) Este fichero define todos los verbos que la librería comprende, tanto en su forma infinitiva como la imperativa, y define, por medio

de las *líneas de gramática*, qué acción va asociada con cada verbo. Aquí se define por ejemplo que el comando «FROTA» causará la acción *Frotar*, lo mismo que los comandos «LIMPIA» y «FRIEGA». El programador puede redefinir estos verbos o añadir otros nuevos, pero aquí están los que la librería comprende por defecto. Al final de este fichero, si no se ha incluido todavía ningún fichero de definición de mensajes de librería, se incluye por defecto `Mensajes.h`.

Mensajes.h (*Nuevo*) Este fichero contiene la definición de los mensajes de librería, que antes se encontraban en `Espanol.h`, en segunda persona. Este es el fichero de mensajes que se incluye por defecto (en `Gramatica.h`).

Msg1P.h (*Nuevo*) Este fichero contiene la definición de los mensajes de librería, que antes se encontraban en `Espanol.h`, en primera persona. Para utilizar este fichero de mensajes hay que incluirlo en el código fuente del juego antes de incluir `Gramatica.h`.

infixe.h (Antes *infix.h*) Este fichero es un módulo de ampliación que proporciona un conjunto de verbos especiales de depuración, con los que se puede realizar una traza de un juego, pudiendo observar y modificar variables, ejecutar rutinas, etc. Para más información al respecto, consulte "The Inform Designer's Manual" (el Manual del Diseñador en Inform) de Graham Nelson.

Facilitar.h (*Nuevo*) El uso de este fichero por parte del programador es totalmente opcional, y su única utilidad es (como su nombre indica) facilitar un poco la lectura y escritura del código fuente del juego, al crear para ello una clase vacía de nombre `Objeto` (con lo que se puede evitar el hacer uso del nombre inglés propio del lenguaje, `Object`) y una clase `Habitacion`, que habilita por defecto el atributo **luz**.

enlazlv.h (Antes *linklv.h*) Este fichero contiene la importación de todas las variables de la librería que son necesarias globalmente. Como ya no se soporta la compilación por módulos, este fichero ha perdido su utilidad y, por tanto, no se distribuye con la librería InformATE!.

enlazlpa.h (Antes *linklpa.h*) Este fichero contiene la declaración de las propiedades y atributos definidos por la librería que ya se encontraban en la librería Inform original. Este fichero se encuentra ahora dentro de `EPARSER.h`.

Los dos últimos eran los que permitían compilar separadamente la librería y el juego, haciendo uso de lo que en Inform se denomina *módulos*. Sin embargo, un bug en la presente versión del compilador de Inform impide que se pueda realizar la compilación separada si se quieren usar también directivas `Zcharacter`. Ya que InformATE! usa estas directivas, se recomienda no forzar la compilación por módulos.

Capítulo 3

Atributos

Un atributo es una bandera asociada con un objeto, que sólo puede tomar los valores **true** (verdadero) o **false** (falso). Por defecto todos los objetos tendrán todas las banderas que a continuación se listan con valor **false**. El programador especificará al crear el objeto qué banderas deben inicializarse a **true** para ese objeto concreto, quedando las restantes (las no especificadas) con su valor **false**.

Los atributos que son definidos y manejados por defecto con la librería InformATE! son, por orden alfabético:

abierto (Antes *open*) Indica si el objeto está abierto. Para la librería sólo tiene sentido en **recipientes** o **puertas**. Admite el sinónimo **abierta**.

abrible (Antes *openable*) Indica si el objeto puede abrirse y cerrarse mediante las acciones *Abrir* y *Cerrar*. Para la librería sólo tiene sentido en **recipientes** o **puertas**.

animado (Antes *animate*) Indica que el objeto en cuestión está vivo. Esto permite que el jugador intente ciertas acciones sobre él, que de otro modo serían rechazadas (por ejemplo, hablar, besarle, etc.). A la vez impide otras acciones (empujarle, tomarle, etc.). Un objeto **animado** habitualmente también proporcionará la propiedad *vida*.

ausente (Antes *absent*) Se usa en objetos que tengan la propiedad *esta_en*. Estos objetos aparecen en muchos lugares simultáneamente y normalmente tienen desactivado el atributo **ausente** ya que cuando se activa, el objeto deja de aparecer en el juego (aunque el objeto en sí no es destruido, por lo que se puede hacer que vuelva a aparecer más tarde).

banderaux (Antes *workflag*) Es un atributo usado internamente por el parser cuando hace listas de objetos (como en los inventarios). El programador no necesita usarlo nunca. Tiene como *alias* **workflag**, por necesidad del compilador.

cerrojo (Antes *lockable*) Este objeto puede ser cerrado con pestillo o usando una llave. Para el primer caso el jugador deberá teclear algo como «ECHA EL PESTILLO A LA PUERTA», y no se requieren objetos extra. En el segundo caso el jugador dirá

«CIERRA LA PUERTA CON LA LLAVE DORADA», y necesitará tener esa llave. Si la puerta en cuestión necesita llave o no la necesita lo decide la propiedad *con_llave*.

El hecho de abrir o cerrar con llave un objeto de este modo, no lo abre o cierra realmente, sino que simplemente cambia el estado de su atributo **cerrojoechado**. Una vez quitado el pestillo será necesario que el jugador abra realmente la puerta con «ABRE PUERTA», para poder pasar. Esto puede ser un poco tedioso, por lo que el programador podrá redefinir la acción *QuitarCerrojo* para que a continuación genere la acción *Abrir*.

cerrojoechado (Antes *locked*) En un objeto con **cerrojo**, este atributo mantiene su estado (si tiene el cerrojo echado o no). Si el cerrojo está echado, una simple orden de abrir el objeto no funcionará (el jugador necesitará antes quitarle el pestillo o abrirlo con la llave adecuada).

comestible (Antes *edible*) El objeto puede comerse. Si no se evita capturando la acción *Comer*, cuando el jugador intente comerlo la librería simplemente lo hará desaparecer y emitirá un mensaje como, por ejemplo, “*Te comes la tortilla. No está mal*”.

conmutable (Antes *switchable*) El objeto puede encenderse y apagarse (independientemente de que emita luz o no). La acción de encender o apagar este tipo de objetos simplemente se traduce en que su atributo **encendido** cambia de valor.

encendido (Antes *on*) Este atributo indica que el objeto está encendido, aunque esto no significa nada para la librería. Es el código del juego el que puede consultar este atributo para decidir qué ocurre si el objeto está encendido o apagado. El valor del atributo puede cambiar si el jugador usa las acciones *Encender* o *Apagar* (siempre que el objeto sea además **conmutable**). Admite también el sinónimo **encendida**.

entrable (Antes *enterable*) El jugador puede *Entrar* en este objeto, aunque el concepto de entrar es bastante vago. Puede tratarse de una silla (con lo que el jugador *entraría* en ella usando el comando «SIÉNTATE EN LA SILLA»), puede ser un vehículo (y entonces el comando sería «ENTRA EN EL COCHE» o «SÚBETE AL COCHE»), etc. Sea como sea, si el objeto no tiene este atributo la librería rechazará esos verbos.

escenario (Antes *scenery*) Este es un atributo útil. Los objetos que lo poseen forman parte del escenario, es decir, el jugador no puede (por defecto) intentar nada con ellos, salvo examinarlos. Por supuesto, los objetos pueden capturar cualquier acción para permitir que ocurra algo distinto del *defecto*, o mostrar algún otro mensaje específico del objeto. Si un objeto tiene este atributo activado, por defecto no se podrá coger y no será mencionado en las descripciones de las localidades donde se encuentre.

estatico (Antes *static*) Similar a **escenario**, el objeto que posea este atributo no puede cogerse ni moverse, pero en cambio sí es listado en las descripciones de las habitaciones (con el típico mensaje “*Puedes ver una estatua*”, por ejemplo). También admite el sinónimo **estatica**.

femenino (Antes *female*) Indica que el nombre corto de este objeto es femenino, de cara a la concordancia con los adjetivos y participios que la librería deba imprimir. Por ejemplo, si el jugador intenta «ABRIR LA CAJA» (suponiendo que tenga los atributos **abrible** y **abierto** activados), la librería responderá “Ya estaba abierta”, en lugar de “Ya estaba abierto”. La decisión la toma comprobando este atributo del objeto *caja*. También se admite el sinónimo **femenina**.

Normalmente este atributo también afecta al artículo que se pone delante del nombre del objeto, pero esto puede no ser siempre así, si el objeto define además la propiedad *genero* (nueva en esta versión española de la librería) o la propiedad *articulos*.

general (Antes *general*) Este atributo no es usado por la librería para nada. Se deja para que el programador lo use en la forma que quiera. Normalmente se usa como bandera indicadora de que el jugador ya ha intentado algo especial con ese objeto, por ejemplo, si el jugador «EXAMINA EL OBJETO» por primera vez, puede indicársele que encuentra algo especial, pero las veces siguientes no queremos repetir el descubrimiento. En este caso el atributo **general** sería la bandera que indica si el jugador ya ha examinado o no el objeto. Inicialmente, al igual que las demás, esta bandera vale **false** para todos los objetos.

hablable (Antes *talkable*) Si el objeto tiene este atributo se permite que el jugador intente verbos para hablar con él. El objeto no tiene por qué representar a una persona o ser animado, sino simplemente algo con lo que se pueda hablar, como típicamente podría ser un micrófono.

luz (Antes *light*) Si el objeto es una habitación, este atributo indica que el jugador puede ver dentro de ella, pues la habitación por sí misma está iluminada (si no tuviera este atributo sólo verá oscuridad, a menos que haya en ella algún objeto que emita luz). Si se trata de otro tipo de objeto, indica que este emite luz. Inform trata con bastante cuidado las reglas de propagación de la luz, de modo que si el jugador entra en una habitación oscura portando un objeto luminoso podrá ver, pero si guarda este objeto luminoso dentro de un **recipiente no transparente**, dejará de ver.

masculino (Antes *male*) Este atributo no es usado por la librería. Un objeto que no tenga el atributo **femenino** se considera automáticamente **masculino**. El atributo se mantiene por compatibilidad con el parser, pero en realidad no se tiene en cuenta y el programador no necesita usarlo nunca.

movido (Antes *moved*) Este atributo está originalmente a **false** para todos los objetos, pero la librería lo pone automáticamente a **true** en un objeto determinado tan pronto como el jugador lo coja (y aunque vuelva a dejarlo su valor se mantendrá). El programador puede consultar este atributo para saber si el objeto ha sido poseído por el jugador en alguna ocasión o no.

La librería lo usa para cambiar la descripción del objeto (ver propiedad *describir*) y para la acción *Objetos* (presente si no se ha definido la constante **NO_LUGARES**).

neutro (Antes *neuter*) No es usado por la librería española. Se mantiene por compatibilidad con el parser, que es capaz de manejar idiomas en los que existe el género neutro (como el inglés, que es el idioma de la librería original), pero el programador de InformATE! no necesita usarlo nunca.

nombreplural (Antes *pluralname*) Un objeto cuyo nombre corto tenga forma plural (aunque el objeto sea único) debe usar este atributo. Por ejemplo, en unas tijeras o unos pantalones. Este atributo, en combinación con **femenino** es usado por la librería para hacer concordancia de género y número en sus mensajes. Por ejemplo, suponiendo que el objeto *tijeras* tenga **abrible**, **abierto**, **femenino** y **nombreplural**, cuando el jugador pida «ABRE LAS TIJERAS», la librería responderá “Ya estaban abiertas”. El mismo mensaje cambia de forma según los atributos de género y número.

En algunos casos este atributo también influye en el artículo que la librería pone delante del nombre, pero esto puede alterarse por la presencia de la propiedad *genero* (nueva en la versión española de la librería) o la propiedad *articulos*.

nombreusado (*Nuevo*) Este atributo lo usa internamente la librería española para marcar cuándo el jugador ha usado el nombre del objeto para referirse a él y cuándo no (ya que el jugador puede referirse a un objeto por su nombre o sólo por su adjetivo como, por ejemplo, en «COGE LA DE MADERA»). Si se usa el nombre o no se usa tiene su importancia a la hora de que el parser decida entre varios objetos que comparten las palabras de vocabulario especificadas por el jugador (como, en el ejemplo anterior, si hubiera una *cuchara de madera*, con *nombre* ‘cuchara’ y *adjetivos* ‘madera’, y algo de *madera carcomida*, con *nombre* ‘madera’ y *adjetivos* ‘carcomida’, el parser elegiría la cuchara).

El programador no debe tocar este atributo.

oculto (Antes *concealed*) Un objeto con este atributo no es listado en la descripción de la habitación. Sin embargo está ahí, y el jugador puede referirse a él (está *al alcance*). El uso habitual es para los objetos que ya se han mencionado en la descripción de la habitación y que por lo tanto no se desea que vuelvan a ser nombrados separadamente. Por ejemplo, con este atributo podríamos evitar la redundancia en “Estás en un salón con una gran chimenea. Puedes ver una gran chimenea”. La última frase de la descripción (que es generada por la librería) se evitaría si la *chimenea* tuviera el atributo **oculto** (o bien su sinónimo **oculta**).

prenda (Antes *clothing*) El jugador puede *ponerse* o *quitarse* objetos que posean este atributo. Cuando se ponga un objeto que sea prenda, éste ganará el atributo **puesto** y aparecerá así indicado en el inventario del jugador (además de que no ocupará sitio en sus manos, dejando lugar para que pueda llevar un objeto más, ver la constante **LLEVAR_MAX**).

propio (Antes *proper*) Lo tienen aquellos objetos cuyo nombre corto sea un nombre propio. La presencia de este atributo en un objeto impide que la librería ponga un artículo definido delante de su nombre en los mensajes, y sustituye el artículo indefinido por “a”. Con esto obtendremos mensajes como “Puedes ver a Manolo”,

en lugar de “*Puedes ver un Manolo*”. El usuario puede cambiar estos artículos por defecto usando la propiedad *articulos* del objeto en cuestión.

puerta (Antes *door*) Los objetos con este atributo son puertas, es decir, objetos que al ser *entrados*, llevan a otro lugar en el juego (no han de ser necesariamente una puerta en el sentido literal, podrían ser **puertas** también un puente o un teletransportador). La implementación completa de un objeto puerta exige que también sean definidas sus propiedades *puerta_a* y *direcc_puerta*. Además una puerta puede tener los atributos **abrible**, **abierto**, **cerrojo** y **cerrojoechado**.

puesto (Antes *worn*) En un objeto que tiene además el atributo **prenda**, el atributo **puesto** indica si la prenda en cuestión la lleva puesta el jugador actualmente o no. Admite el sinónimo **puesta**.

recipiente (Antes *container*) Este atributo indica que se pueden meter otros objetos dentro de este. El recipiente puede además ser **abrible**, y estar inicialmente **abierto** (o cerrado, esto es \sim abierto), o tener **cerrojo** y estar inicialmente con **cerrojoechado** (o no).

soporte (Antes *supporter*) Indica que se pueden poner objetos encima de este, o incluso el jugador puede intentar subirse encima de él.

Es importante tener en cuenta que un objeto no puede ser **recipiente** y **soporte** a la vez, debido a la forma en que son implementados por Inform. En ambos casos, todos los objetos que el jugador meta dentro del recipiente o ponga encima del soporte, se almacenarán internamente como objetos contenidos. La única diferencia es que cuando la librería deba mostrar los contenidos de un **recipiente** dirá algo como “*En la caja puedes ver...*”, pero si se tratara de un **soporte** diría “*Sobre la caja puedes ver...*”. Esta es probablemente una deficiencia de la librería arrastrada desde sus primeras versiones.

transparente (Antes *transparent*) Los objetos *transparentes* permiten que los objetos contenidos sean visibles, y que la luz pase a través de ellos. Pueden ser **recipientes** o no. Por ejemplo, una radio puede contener al objeto *dial*, para que vayan juntos a todos lados, pero la radio no es un **recipiente** dentro del cual está el dial. Sin embargo, hay que hacer la radio **transparente** para que el jugador pueda referirse al dial, que está internamente almacenado como parte de la radio. De lo contrario el dial no sería visible por no estar dentro de un **recipiente abierto**.

Si se coloca un objeto luminoso (con el atributo **luz**) dentro de un **recipiente** cerrado, pero siendo este recipiente **transparente**, la luz puede todavía iluminar la habitación.

valepuntos (Antes *scored*) Los objetos que tienen este atributo hacen que la puntuación del jugador aumente al ser recogidos o, si se trata de un lugar, cuando el jugador llega a ellos. La puntuación aumenta en la cantidad **PUNTOS_OBJETO** o **PUNTOS_LUGAR** respectivamente, pero esto sólo ocurrirá la primera vez en que son cogidos o visitados.

visitado (Antes *visited*) Este atributo lo tienen los objetos que representan lugares y es usado por la librería para saber si el jugador ya ha estado antes en ese sitio. Esto afectará a la descripción del lugar, que no se mostrará la próxima vez que llegue a él si *modomirar* es distinto de 2, y también a la puntuación por visitar el lugar, que sólo se sumará la primera vez que se visite. También es utilizado por la acción *Lugares*, presente si no se ha definido la constante **NO_LUGARES**.

Normalmente el programador no necesita usar este atributo, salvo para consultarlo para ver si el jugador ya ha estado o no en ciertos lugares del juego (por ejemplo, para no darle pistas sobre lugares por los que todavía no ha pasado).

Capítulo 4

Propiedades

Una propiedad es un dato asociado con un objeto. A diferencia de los atributos que sólo pueden tomar los valores *verdadero* o *falso*, las propiedades son mucho más flexibles, ya que pueden contener:

- Un número entero (por ejemplo, el número de objetos que caben como máximo dentro de una caja sería una propiedad de esta caja).
- Una cadena de caracteres (por ejemplo, el nombre corto de esta caja, algo como "caja de plástico", o su descripción).
- Una lista de palabras separadas por espacios (por ejemplo, todos los sinónimos que el jugador podría usar para referirse a este objeto), que han de ir entre comillas simples.
- Una rutina (también llamada función o método, esta última designación sólo en el caso de que se trate de una propiedad en un objeto). Es un código que se ejecutará cuando se lo pidamos. Por ejemplo, una caja podría tener un método llamado *peso* que calcule cuánto pesa la caja, analizando para ello todo lo que contiene. Desde cualquier otro objeto podríamos llamar a `caja.peso()` para que la caja nos diga cuánto pesa.

Cualquier propiedad puede contener cualquiera de los tipos anteriores. No hay tipos asociados con propiedades, pero la librería busca en algunas propiedades como si fueran listas, mientras que ejecuta otras como si fueran métodos. Por tanto si no ponemos en la propiedad lo que la librería espera aparecerán errores.

En la siguiente lista se indica qué debe haber en cada propiedad de las que vienen por defecto con la librería y para qué se usan.

abajo (Antes *d_to*) Ver el epígrafe *direcciones*.

adentro (Antes *in_to*) Ver el epígrafe *direcciones*.

adjetivos (*Nuevo*) Contiene una lista de palabras de diccionario que el jugador puede usar como adjetivos para referirse a este objeto. Por ejemplo, para el objeto *llave de hierro*, un adjetivo sería 'hierro' (la preposición *de* es automáticamente ignorada por el parser cuando se utiliza para nombrar un objeto). El jugador podrá también especificar objetos usando sólo su adjetivo. Por ejemplo, «COGE HIERRO» o «COGE LA DE HIERRO», sería entendido como una referencia a este objeto. Sin embargo, si hay otro objeto disponible cuyo *nombre* coincida con este adjetivo, entonces este último será preferido en tal situación.

Veamos otro ejemplo: si en una misma localidad coinciden los objetos *llave de hierro* (with nombre 'llave', adjetivos 'hierro') y *hierro retorcido* (with nombre 'hierro', adjetivos 'retorcido'), cuando el jugador ponga «EXAMINA HIERRO», el parser entenderá que se refiere al hierro retorcido, ya que ha encontrado la palabra 'hierro' en el nombre de ese objeto (y entre los adjetivos de otro), y la propiedad *nombre* siempre tendrá prioridad sobre la propiedad *adjetivos* en la desambiguación.

Esto es una novedad de la librería española. En la librería original no se permitía separar adjetivos, sino que estos se listaban como parte de la propiedad *nombre*. Así la llave de hierro tendría como nombre dos palabras 'llave' y hierro, y el hierro retorcido otras dos 'hierro' y 'retorcido'. Un intento de «EXAMINAR HIERRO» sumiría al parser en la confusión ya que la palabra 'hierro' aparece en ambos objetos. El parser preguntaría al jugador: “¿A cuál te refieres exactamente, a la llave de hierro o al hierro retorcido?”. Esto no era muy inteligente.

afuera (Antes *out_to*) Ver el epígrafe *direcciones*.

al_e (Antes *e_to*) Ver el epígrafe *direcciones*.

al_n (Antes *n_to*) Ver el epígrafe *direcciones*.

al_ne (Antes *ne_to*) Ver el epígrafe *direcciones*.

al_no (Antes *nw_to*) Ver el epígrafe *direcciones*.

al_o (Antes *w_to*) Ver el epígrafe *direcciones*.

al_s (Antes *s_to*) Ver el epígrafe *direcciones*.

al_se (Antes *se_to*) Ver el epígrafe *direcciones*.

al_so (Antes *sw_to*) Ver el epígrafe *direcciones*.

antes (Antes *before*) Esta propiedad es fundamental para que el juego haga algo distinto de lo que la librería supone por defecto. Se trata de una rutina (o método) que es llamada por la librería una vez que el comando del jugador ha sido comprendido, informando al objeto de lo que el jugador pretende hacer (por ejemplo «ABRE CAJA» causará que se llame a esta rutina para el objeto *caja*, informándole de que la acción solicitada es *Abrir*).

El objeto tiene aquí la oportunidad de emitir algún mensaje especial o cambiar cualquier otra variable del juego. Si no lo hace, la librería continuará y aplicará su método estándar (es decir, en el ejemplo se trataría de comprobar si el objeto es **abrible** y, si no lo es, emitir un mensaje de error, y si lo es comprobar que no esté ya **abierto**, o que no tenga **cerrojo**echado, etc.)

De este modo, cualquier objeto puede sustituir localmente las acciones por defecto de la librería. Insistimos en que esta rutina es llamada para el objeto involucrado *antes* de que la librería intente llevar a cabo la acción. Si el comando del jugador tiene dos objetos (por ejemplo «ABRIR CAJA CON LLAVE AMARILLA») sólo se llama a la rutina *antes* del objeto directo (*caja*), pero la variable *otro* tiene el valor del otro objeto (la llave amarilla).

Si el objeto es una habitación, su rutina *antes* será llamada siempre para todas las acciones que el jugador intente en esa habitación, antes incluso de llamar a la rutina *antes* del objeto involucrado. Así, por ejemplo podemos cambiar la respuesta estándar “No hueles nada extraño” ante la acción *Oler* para que en una habitación concreta sea distinta. O podemos hacer habitaciones en las que ciertos comandos no funcionen o funcionen de forma diferente.

La rutina no recibe parámetros, y ha de tener una estructura similar a la de un *switch*, en el que cada opción sería el nombre de una acción y el código a ejecutar para esa acción concreta.

El método *antes* debe devolver **true** o **false**. Es como si la librería le preguntara: ¿Te has ocupado tú de esta acción?. Si la rutina responde **true**, entonces la librería ya no hace nada más. Si responde **false**, la librería se hace cargo de esa acción ejecutando las rutinas por defecto.

arriba (Antes *u_to*) Ver el epígrafe *direcciones*.

articulo (Antes *article*) Contiene una cadena de caracteres que es el artículo indefinido más adecuado para este objeto. Por defecto vale "un", pero realmente la librería española averigua uno mejor según el género y número del objeto. La librería puede elegir "un", "una", "unos" o "unas" o "a" para los nombres propios. Si ninguno de estos es adecuado para este objeto, se puede especificar aquí uno mejor.

El uso típico es para objetos que es mejor que no lleven artículo indefinido. Por ejemplo “Aquí hay un agua”, queda muy mal y es preferible “Aquí hay agua” o “Aquí hay un poco de agua”. En estos casos basta con definir la propiedad *articulo* del objeto agua dándole el valor "" (cadena vacía) en el primer caso, o bien "un poco de" en el segundo caso.

articulos (Antes *articles*) Es similar a la propiedad *articulo*, sólo que en vez de redefinir únicamente el artículo indefinido, redefine también el artículo definido. Contiene una lista con tres cadenas de caracteres. La primera cadena será el artículo definido con la inicial en mayúscula, la segunda será el artículo definido (normal) y la tercera el artículo indefinido.

Normalmente no es necesario usar esta propiedad, ya que la librería decide inteligentemente si debe usar "el", "la", "los", "las", etc. según el género y

número del objeto. Pero si no resultara adecuada la elección, puede cambiarse para un objeto concreto usando esta propiedad.

cada_turno (Antes *each_turn*) Esta propiedad es evaluada al final de cada turno para todos los objetos al alcance del jugador (después incluso de haber ejecutado todos los relojes y *daemons*). Puede contener una cadena de caracteres (que será impresa en pantalla por la librería) o una rutina (que será ejecutada en ese momento).

cantidad (Antes *number*) Es una propiedad de propósito general, que la librería no usa para nada. El programador es libre de usarla para guardar en ella cualquier número que guarde alguna relación con el objeto (por ejemplo, para un teléfono podría guardar el número de teléfono, aunque está limitado a un valor máximo de 32.767).

Excepción: los objetos que representen al jugador (por defecto es el objeto **objjugador**, pero esto puede cambiarse), deben tener esta propiedad declarada con un valor inicial de 0, y no deben usarla para nada.

capacidad (Antes *capacity*) Esta propiedad contiene un número (por defecto 100) que representa el número de objetos que caben dentro o sobre éste (suponiendo que sea un **recipiente** o un **soporte**). También puede ser una rutina que devuelva como resultado un número.

Para el jugador esta propiedad representa el número de objetos que puede llevar (en las manos). En este caso su valor inicial es el de la constante **LLEVAR_MAX**, para el objeto **objjugador**, que es el objeto que hace de jugador por defecto.

con_llave (Antes *with_key*) En los objetos que tienen el atributo **cerrojo**, esta propiedad indica qué llave es la que sirve para quitarles el cerrojo. Si la propiedad **con_llave** no existe, no es necesaria llave (basta que el jugador ordene «QUITA EL PESTILLO A ...»), pero si existe, el objeto especificado en ella será el que el jugador tiene que usar en la orden «ABRE ... CON ...».

daemon (Antes *daemon*) Esta rutina es ejecutada al final de cada turno, pero sólo a partir del momento en que haya sido activada con la función **ArrancarDaemon**, y dejará de ser llamada cuando se desactive con la función **PararDaemon**.

Esta rutina es la clave para permitir que los PNJ (*Personajes No Jugadores*) se muevan o hagan cosas según pasa el tiempo de juego.

describir (Antes *describe*) Esta propiedad es una rutina. En las habitaciones es llamada por la librería justo antes de imprimir la *descripcion* larga de la habitación. En los objetos, es llamada cuando la librería ha impreso la descripción de la habitación y se dispone a imprimir la parte de “Puedes ver...”. Antes de esto, la librería llama a la rutina *describir* para cada objeto contenido en la habitación, como si le preguntara: *¿Te ocupas de describirte tú mismo?*. Si el objeto retorna **true**, entonces la librería no lo mencionará en la descripción adicional. Si retorna **false**, sí lo hará. El objeto puede aprovechar esta rutina para mostrar una descripción de sí mismo levemente distinta de la que daría la librería. Por ejemplo “Aquí hay una ruidosa lavadora que

parece estar en su programa de centrifugado”. Naturalmente esta rutina puede consultar atributos y propiedades del objeto para construir una descripción adecuada (cosa que la librería no hará, a menos que se trate de atributos estándar).

Ver también *si_encendido*, *si_apagado*, *si_abierto* y *si_cerrado*.

descripcion (Antes *description*) Para una habitación es la descripción larga de la misma (la que se da la primera vez que el jugador entra en ella, o cada vez que entra en ella si *modomirar* vale 2, o cuando se da la orden «MIRAR»). Para un objeto es la descripción que se dará cuando el jugador dé la orden «EXAMINAR OBJETO». En ambos casos puede ser simplemente una cadena de caracteres (que será impresa por la librería), o una rutina que se ocupe de imprimir un mensaje adecuado (en este caso la librería no tendrá en cuenta el valor que retorne dicha rutina).

descripcion_dentro (Antes *inside_description*) Sólo puede usarse en objetos que tengan el atributo **entrable**. En este caso, si el jugador está dentro del objeto y pide «MIRAR», se imprimirá esta propiedad como parte de la descripción de la habitación (en párrafo aparte).

despues (Antes *after*) Esta propiedad es una rutina, muy similar a *antes* en cuanto a su estructura y utilidad. Es llamada por la librería una vez que la acción ha tenido lugar, pero antes de informar al jugador de ello (es decir, antes de que la librería haya escrito ningún mensaje). Se llama primero a la rutina *despues* de la habitación donde ha ocurrido la acción, y a continuación a la rutina *despues* del objeto directo de la acción.

Al igual que en el caso de la propiedad *antes*, si la rutina retorna **true**, la librería no continúa, y si retorna **false**, la rutina sigue con su procedimiento y muestra el mensaje estándar (que será algo del estilo de “*Hecho*”).

Por ejemplo, imaginemos una joya que al cogerla activa una alarma. La rutina *despues* de este objeto contendría el código que active esa alarma y tal vez un mensaje para avisar al jugador de ello. Es importante poner este código en la rutina *despues*, porque así nos aseguramos de que realmente el jugador ha cogido el objeto (pensemos que aunque el jugador ponga «COGER JOYA», puede ser que la acción fracase si la joya no está a su alcance, sin embargo, si llegamos a la rutina *despues* es porque realmente el jugador ha cogido la joya).

direcciones No hay ninguna propiedad con este nombre, pero en este epígrafe agrupamos la explicación de las propiedades *al_n*, *al_s*, *al_e*, *al_o*, *al_ne*, *al_no*, *al_se*, *al_so*, *adentro*, *afuera*, *arriba* y *abajo*.

Todas estas propiedades pueden formar parte de un objeto que represente una localidad, y son evaluadas cuando el jugador intenta moverse en la dirección correspondiente (norte, sur, este, oeste, nordeste, noroeste, sureste, suroeste, entrar, salir, subir y bajar, respectivamente).

El valor de la propiedad puede ser:

- Vacío (si la propiedad no aparece en la lista de propiedades para esa habitación). En este caso la librería imprimirá el mensaje estándar “*No puedes ir por ahí*”.
- Una cadena de caracteres. En este caso la librería imprimirá esa cadena de caracteres cuando el jugador intente usar esa salida, y no ocurrirá movimiento alguno. Es una forma de cambiar el mensaje estándar para algunas salidas.
- Un objeto habitación. En este caso el jugador será movido a esa habitación (vamos, lo normal).
- Un objeto **puerta**. En ese caso la librería comprobará el atributo **abierto** de la puerta para ver si está cerrada (en cuyo caso no le dejará pasar) o si está abierta (en cuyo caso evaluará la propiedad *puerta_a* de la puerta para averiguar a dónde lleva). Las puertas son objetos un poco complicados como ya se verá.
- Una rutina. En ese caso la rutina puede imprimir el texto que quiera y hacer las averiguaciones que quiera, antes de retornar finalmente **false** si decide que el jugador no puede pasar por esa dirección, o bien un objeto lugar, en el caso de que pueda pasar (y entonces indica el lugar donde aparecerá).

direcc_puerta (Antes *door_dir*) Sólo puede usarse en objetos que tengan el atributo **puerta**. En este caso indica en qué salida de la habitación se halla la puerta y puede tomar el valor de una dirección cualquiera (por ejemplo *al_s*). De este modo, si el jugador escribe «VE HACIA LA PUERTA» o «PASA POR LA PUERTA» o similar, la librería examinando la propiedad *direcc_puerta*, convierte esa acción en la acción «IR AL SUR». A su vez la habitación deberá tener en su salida *al_s* a la puerta en cuestión. Esto es un poco enrevesado, pero es la forma en que funciona la librería.

La propiedad *direcc_puerta* puede ser una rutina, que tiene que devolver una dirección como *al_n*, *al_s*, etc. Es habitual que sea una rutina en las puertas bidireccionales, por ejemplo, una puerta que conecta un pasillo (al norte) con un vestíbulo (al sur). La puerta se halla al norte del vestíbulo, pero al sur del pasillo, por lo que la propiedad *direcc_puerta* ha de ser una rutina que examine la variable **localizacion**, para saber dónde está el jugador, y si está en el vestíbulo retornará *al_n*, mientras que si está en el pasillo retornará *al_s*.

Véase también *puerta_a*.

Nota adicional: Existe un objeto en la librería, llamado **brujula**, que usa de forma diferente esta propiedad. Ver el epígrafe del objeto **brujula** para más detalles.

esta_en (Antes *found_in*) Algunos objetos pueden aparecer en muchas habitaciones a la vez. Por ejemplo un río (como el de la aventura original) puede cruzar varias localidades, o una puerta puede aparecer en dos habitaciones a la vez, o quizás un castillo lejano es visible desde varios lugares. Para casos como estos se tiene esta propiedad, que está formada bien por una lista de localidades, o bien por una rutina.

Si se trata de una lista de localidades, el objeto será accesible desde todas las localidades listadas. Si es una rutina, la librería la llamará pasándole como parámetro un objeto localidad, y la rutina debe responder **true** si el objeto es accesible desde ese

lugar, o **false** si no lo es. Si se hace una rutina que responde siempre **true** se tendrá un objeto accesible desde todos los lugares del juego (por ejemplo un suelo).

Nota: En realidad el objeto no está en todos esos lugares a la vez, sino que cada vez que el jugador se mueve de un lugar a otro, la librería mueve este tipo de objetos, para que siempre estén en la misma habitación que el jugador (suponiendo que dicha habitación pase el test de *esta_en*).

genero (*Nuevo*) Esta propiedad, exclusiva de la librería española, contiene el género y número de la cadena que se imprimirá como nombre de este objeto. Este género y número se expresa mediante un número entero, cuyos valores posibles son:

- 0 Masculino singular (este es el valor por defecto)
- 1 Femenino singular
- 3 Masculino plural
- 4 Femenino plural

Para no tener que memorizar estos números que a simple vista carecen de significado, podemos hacer uso de las constantes **G_MASCULINO**, **G_FEMENINO** y **G_PLURAL** para obtener dichos valores. El modo de usarlas es muy sencillo, basta con sumar aquellas que representen el género y número del objeto en cuestión. Por ejemplo, **G_MASCULINO** + **G_PLURAL** indicaría género masculino y número plural, y **G_FEMENINO** indicaría género femenino y número singular (no hay ninguna constante que indique el género singular, pues no es necesaria).

La librería usará esta propiedad para decidir qué artículo ha de poner delante de la palabra en cuestión. Si la propiedad no está presente (o vale 0), usará en cambio los atributos **femenino** y **nombreplural**. Puede parecer que estamos replicando innecesariamente la información ya que, si podemos darla en forma de atributos, ¿por qué darla en forma de una propiedad adicional?

La razón estriba en que la propiedad **genero** es un valor constante, mientras que los atributos **femenino** y **nombreplural** pueden ser activados y desactivados varias veces por la librería, perdiéndose entonces el género y número del objeto.

Todo esto es debido a que para referirse a un objeto, la librería usa siempre un mismo nombre (el que el programador le haya dado), por ejemplo `nombre_corto` "baúl", cuyo género y número es invariable (la palabra *baúl* siempre es masculina singular), sin embargo el jugador para referirse a ese mismo objeto podría usar una gran variedad de sinónimos (nombre 'baul' 'arca' 'arcon' 'cofre' etc.) y algunos son masculinos y otros femeninos. La librería española detecta si la palabra usada por el jugador es masculina o femenina, y entonces cambia el atributo **femenino** de ese objeto. Si, por ejemplo, el jugador dice «ABRE ARCA», la librería activará el atributo **femenino** para ese objeto, mientras que si dice «ABRE COFRE», desactivará ese atributo. De este modo los mensajes por defecto de la librería siempre concordarán en género y número con la palabra usada por el jugador (por ejemplo, ante «ABRE ARCA», responderá "Ya estaba abierta" y, en cambio, ante «ABRE COFRE» responderá "Ya estaba abierto").

Así que los atributos **femenino** y **nombreplural** van variando según las palabras que use el jugador. Por eso es necesaria otra propiedad que guarde el género y número del *nombre_corto* del objeto pues, de lo contrario, el artículo que la librería ponga delante de él sería “la” o “el” dependiendo de cómo se hubiera referido el jugador a este objeto por última vez. Por ejemplo: «ABRE ARCA», “Abres la baúl”. Insistimos en que el nombre del objeto que imprime la librería es siempre “baúl”, aunque el jugador pueda usar diferentes sinónimos.

Si el objeto no tiene sinónimos de diferentes géneros o números, entonces sus atributos **femenino** y **nombreplural** no cambian nunca, por lo que no es necesario definir su propiedad *genero*. En cambio, si tiene sinónimos de géneros o números diferentes, el programador deberá darle a la propiedad *genero* el valor adecuado, según el género y número del nombre *principal* (el nombre corto). Si se omite esta propiedad, la librería mantendrá siempre el género y número dado por los atributos **femenino** y **nombreplural**, que no serán modificados nunca (aunque el jugador use sinónimos de diferente género y número).

Véanse también las propiedades *nombre_m*, *nombre_f*, *nombre_mp*, *nombre_fp* y las constantes **G_MASCULINO**, **G_FEMENINO** y **G_PLURAL**.

gramatica (Antes *grammar*) Esta oscura propiedad es una rutina que puede darse sólo en los objetos con los que el jugador puede hablar (es decir, los que tienen el atributo **animado** o **hablable**). Normalmente no será necesario programar esta propiedad (¡afortunadamente!, porque es bastante complicado).

Veamos cómo funcionaría: Para dar órdenes o hablar con estos objetos el jugador pondrá algo como «MANOLO, ABRE LA CAJA». El parser, una vez que ha determinado que se trata de una orden para el objeto *Manolo*, llamará al procedimiento *gramatica* de este objeto, el cual prácticamente ha de ser otro parser que interprete el resto del comando.

El procedimiento *gramatica* se encontrará con ciertas variables ya prefijadas por el parser, en particular la variable *palabra_verbo* contiene la palabra de diccionario usada por el jugador (‘abre’ en nuestro ejemplo), si es que esa palabra estaba en el diccionario, ó 0 si no lo estaba. Por otro lado, la variable *palabra_verbonum* indica qué número ocupa esa palabra entre las palabras tecleadas por el jugador (en nuestro caso sería 3, ya que la palabra 1 es Manolo, la 2 es la coma y la 3 es el verbo en cuestión).

A partir de aquí, la rutina *gramatica* del objeto *Manolo* puede si lo desea intentar descifrar el resto del comando, y alterar la variable *palabra_verbonum* para saltarse palabras. Además, si descifra correctamente todo el comando cargará en la variable **accion** el número de la acción detectada, en la variable **uno** el objeto directo sobre el que se aplica la acción (en nuestro caso sería el objeto *caja*, si es que hay tal cosa al alcance de Manolo) y en la variable **otro** el otro objeto si lo hay (en este caso no lo hay). Cuando haya terminado retornará: **true** para indicar a la librería que toda la interpretación ha terminado y que ya tiene en las variables **accion**, **uno** y **otro** los resultados; **false** para indicar que mejor lo descifre todo la librería; una palabra de diccionario como ‘rompe’ por ejemplo, que significa que la interpretación

la haga la librería, pero usando como *palabra_verbo* 'rompe' en vez de la original ('abre'); o finalmente puede retornar el valor negativo de una palabra de diccionario, por ejemplo -'rompe', para indicar que la interpretación la haga la librería, primero probando a ver si puede hacerlo con el verbo 'rompe', y luego, si la gramática de ese verbo no encaja bien con lo que el jugador ha escrito, que lo reintente con el verbo originalmente escrito por él.

Hay que decir que normalmente esta rutina no es necesaria. Si no está presente, la librería se ocupará de descifrar lo que el jugador ha escrito y, si lo logra, cargará adecuadamente las variables *accion*, *uno* y *otro* y llamará a la rutina *ordenes* de *Manolo* para ver si éste quiere llevar a cabo esa acción o no. Si, en cambio, lo que ha escrito el jugador no tiene sentido para la librería, ésta llamará a la rutina *ordenes* de *Manolo* poniendo en la variable *accion* la acción falsa *NoComprendido*, y en la variable *tipoerror* el número de error de parser (así quizás *Manolo* todavía pueda emitir algún mensaje coherente).

imperativo (*Nuevo*) Esta propiedad es exclusiva de la librería en español. En los objetos de la clase *VerboIrregular*, esta propiedad contiene la forma imperativa de dicho verbo. Por ejemplo, el *VerboIrregular* "fregar" tendría en esta propiedad el valor 'friegas'. En realidad, se trata de una lista de palabras de diccionario, en las que la librería buscará el verbo que ha usado el jugador cuando necesite imprimir la forma infinitiva de ese verbo. Así, por ejemplo, si el jugador escribe «EX» (abreviatura de «EXAMINAR») la librería debe responder "¿Qué quieres examinar?". Para lograr esto definimos que el *VerboIrregular* "examinar" tiene como una de sus formas imperativas 'ex'. Si no lo hiciéramos así la librería generaría entonces el mensaje "¿Qué quieres exr?", ya que al no encontrar ese verbo en la lista de verbos irregulares, lo considerará regular y formará su infinitivo simplemente añadiendo una "r".

Esta propiedad es también usada si el jugador escribe sus comandos en infinitivo («FREGAR ESPADA»). En este caso, al no comprender el verbo "fregar" (ya que la gramática sólo define el verbo 'friegas'), la librería buscará en todos los verbos irregulares uno que se llame igual que la palabra usada por el jugador. Encontrará el *VerboIrregular* "fregar" que coincide, y entonces extraerá de la propiedad *imperativo* la correspondiente forma imperativa ('friegas') que sustituirá a lo que había escrito el jugador.

Esta propiedad es en realidad un *alias* de la propiedad *nombre*, por lo que un objeto no puede definir a la vez ambas propiedades (de todas formas esto nunca es necesario).

inicial (Antes *initial*) Es una cadena de texto o una rutina, que es mostrada (o ejecutada) por la librería cuando desea mencionar un objeto que está en una habitación y que todavía no ha sido poseído por el jugador. Esto permite variar el mensaje "Aquí puedes ver una máquina", de forma que sea "Aquí puedes ver una máquina que nadie parece haber tocado en años", por ejemplo, pero sólo hasta que el jugador la recoja. Aunque la deje de nuevo allí, posteriormente la librería ya no usará la propiedad *inicial* para describirla, sino la propiedad *describir* (y si ésta no existe, generará la descripción estándar "Puedes ver una máquina").

Las habitaciones también pueden tener la propiedad *inicial*, y en ese caso es el mensaje que saldrá cuando el jugador entre en la habitación (cada vez que entre), antes de la descripción de la misma (pero no saldrá al «MIRAR» la habitación). Esto resulta útil cuando queramos mostrar un texto (por ejemplo, “*Llegas a un cruce de caminos*”) cuando se entre a la localidad, pero no cuando el jugador pida re-describirla.

irrelevante (*Nuevo*) En los objetos que representan habitaciones esta propiedad mantiene una lista de palabras *irrelevantes*. Si el jugador intenta cualquier acción sobre una palabra de estas, recibirá por defecto el mensaje “*Eso no es importante para el juego*”.

No obstante, si hay un objeto *de verdad* en esa habitación cuyo nombre coincide con una de las palabras listadas, el mensaje anterior no saldrá, sino que la acción tendrá lugar sobre el objeto en cuestión.

Esta propiedad es en realidad un *alias* de la propiedad *nombre*, por lo que no pueden usarse ambas en el mismo objeto. De todas formas, las habitaciones en InformATE! no tienen *nombre* por el cual referirse a ellas.

De hecho, en la librería original de Inform, la lista de palabras irrelevantes se almacenaba en la propiedad *name* de la habitación, pero en mi opinión esto quedaba poco claro, por lo que he decidido darle a esta propiedad el alias *irrelevante* (en realidad sigue siendo lo mismo, pero hace los listados un poco más legibles).

listar_juntos (Antes *list_together*) Esta propiedad es útil cuando tenemos muchos objetos de parecidas características (por ejemplo, varios tipos de comida, o distintos tesoros) y es deseable que aparezcan juntos en el inventario.

Para estos casos se usa esta propiedad. En general, al hacer inventarios se juntarán todos los objetos cuya propiedad *listar_juntos* tenga el mismo valor, y si esta propiedad es una cadena, se imprimirá esta como nombre preliminar del grupo. Así, si los objetos *tarta*, *bizcocho*, *caramelo* y *bombón*, tienen todos ellos la propiedad *listar_juntos* igual a 1, saldrán juntos en los inventarios (o listas de objetos presentes en un lugar). Y si el valor de *listar_juntos* es la cadena “*golosinas*”, el inventario sería algo así como: “*Llevas cuatro golosinas: una tarta, un bizcocho, un caramelo y un bombón*”.

En general, la creación de listados de objetos agradables de leer es muy compleja e Inform proporciona gran cantidad de opciones para ello que se saldrían de los objetivos de este pequeño resumen. El lector interesado debe saber que el manual original dedica un capítulo completo a este asunto.

No obstante, en la mayoría de los juegos no suele haber este tipo de colecciones de objetos similares, por lo que esta propiedad normalmente no es necesaria. Para el caso de que los objetos no sean sólo similares, sino idénticos resulta más sencillo usar la propiedad *plural* (véase).

listarse (Antes *invent*) Cuando la librería crea listas de objetos, añade entre paréntesis algunos detalles sobre los mismos. Por ejemplo, en los objetos **abribles** suele poner

entre paréntesis si está abierto o cerrado, en las **prendas** especifica si la llevas puesta o no, en los **conmutables** aclara si está encendido o apagado.

Esto está bien para los atributos que la librería conoce, pero el programador puede definir atributos y propiedades nuevas en un juego. Por ejemplo, una botella puede tener una propiedad que indique la cantidad de líquido que contiene. Ya que la librería no sabe de esta propiedad, no puede añadir entre paréntesis algo como “(*medio llena*)” al listar la botella. Para esto es para lo que se proporciona esta propiedad.

La rutina *listarse* es llamada por la librería cuando está construyendo una lista de objetos, para dar una oportunidad al objeto de describirse un poco mejor en los inventarios. La discusión completa de esta facilidad sería demasiado extensa (se remite al lector interesado al manual original).

nombre (Antes *name*) Contiene una lista de palabras de diccionario (las cuales han de ir entre comillas simples), separadas por espacios. Se trata de los sinónimos que el jugador puede utilizar para referirse a este objeto.

Si todos los sinónimos tienen el mismo género y número (por ejemplo, ‘falda’ ‘prenda’ ‘ropa’ ‘minifalda’ pueden ser todos ellos sinónimos de un mismo objeto y además todos son del género femenino y número singular), entonces pueden listarse todos bajo la misma propiedad *nombre*, y especificar en el atributo **femenino** si es femenino o no (el no especificar este atributo lo hace masculino).

Sin embargo, si los sinónimos tienen diferentes géneros o números (por ejemplo, ‘pantalón’ ‘pantalones’ ‘ropa’ ‘vaqueros’ son todos sinónimos para un mismo objeto, y sin embargo ‘pantalón’ es masculino singular, ‘pantalones’ es masculino plural, ‘ropa’ es femenino singular, etc.) en ese caso conviene separar la lista de sinónimos en varias listas separadas por género y número y guardarlas en las propiedades *nombre_m*, *nombre_f*, *nombre_mp* y *nombre_fp*, y asignar a la propiedad *genero* el género y número del *nombre corto* (véase la propiedad *nombre_corto* a continuación).

La propiedad *nombre* es en realidad un *alias* de la propiedad *name*, la cual debe seguir existiendo con su nombre en inglés debido a cómo funciona el compilador. No obstante, un programador hispanohablante no necesita usar nunca la propiedad como *name* y usará *nombre* en su lugar.

nombre_corto (Antes *short_name*) Es el nombre corto del objeto (el que usa la librería cuando tiene que nombrar el objeto). Normalmente no hace falta especificar esta propiedad, ya que el nombre corto del objeto será el que le demos entre comillas en la declaración del objeto. No obstante, si queremos que el nombre del objeto cambie (por ejemplo, puede cambiar de “jarrón” a “jarrón roto”), lo mejor es usar esta propiedad y hacer que contenga una rutina que imprima “*jarrón*” o bien “*jarrón roto*” según el estado de alguna bandera (por ejemplo el atributo **general** del jarrón podría servir para esto).

Como se deduce de lo anterior, esta propiedad puede ser una cadena de texto o una rutina. La librería la utilizará cada vez que necesite imprimir el nombre del objeto (en los inventarios, al mencionarlo en una habitación o al mencionarlo en cualquiera de sus mensajes estándar como en “No creo que empujar el jarrón sirva para nada”).

nombre_corto_indef (Antes *short_name_indef*) Es una rutina que ha de imprimir el nombre corto del objeto cuando va precedido de un artículo indefinido. Esto no es útil en el idioma español, ya que el nombre de un objeto no cambia al poner un artículo indefinido, pero está presente para otros idiomas. Lo mejor es no usarla nunca, entonces la librería usará la propiedad *nombre_corto* y si ésta tampoco está definida para el objeto, usará el nombre dado en la declaración del mismo.

nombre_f (*Nuevo*) Lista de sinónimos del género femenino y número singular que son aplicables a este objeto (sólo en objetos que tengan sinónimos con diferentes géneros o números, si no es este el caso pueden ponerse todos estos bajo la propiedad *nombre* y aclarar su género usando el atributo **femenino**).

Cuando se usa la propiedad *nombre_f* conviene también especificar la propiedad *genero* para aclarar el género y número del *nombre corto*.

nombre_fp (*Nuevo*) Lista de sinónimos del género femenino y número plural que son aplicables a este objeto (sólo en objetos que tengan sinónimos con diferentes géneros o números, si no es este el caso pueden ponerse todos estos bajo la propiedad *nombre* y aclarar su género y número usando los atributos **femenino** y **nombreplural**).

Cuando se usa la propiedad *nombre_fp* conviene también especificar la propiedad *genero* para aclarar el género y número del *nombre corto*.

nombre_m (*Nuevo*) Lista de sinónimos del género masculino y número singular que son aplicables a este objeto (sólo en objetos que tengan sinónimos con diferentes géneros o números, si no es este el caso pueden ponerse todos estos bajo la propiedad *nombre*).

Cuando se usa la propiedad *nombre_m* conviene también especificar la propiedad *genero* para aclarar el género y número del *nombre corto*.

nombre_mp (*Nuevo*) Lista de sinónimos del género masculino y número plural que son aplicables a este objeto (sólo en objetos que tengan sinónimos con diferentes géneros o números, si no es este el caso pueden ponerse todos estos bajo la propiedad *nombre* y aclarar su número usando el atributo **nombreplural**).

Cuando se usa la propiedad *nombre_mp* conviene también especificar la propiedad *genero* para aclarar el género y número del *nombre corto*.

no_puedes_ir (Antes *cant_go*) Esta propiedad sólo tiene sentido en los objetos que representan una localidad. Se trata de una cadena o una rutina que debe imprimir un texto, el cual aparecerá cuando el jugador intente ir en una dirección en la que no hay salida (por ejemplo, para imprimir “No parece que por allí haya ninguna salida”, en lugar del mensaje por defecto: “No puedes ir por ahí”).

ordenes (Antes *orders*) Esta propiedad, que es una rutina, sólo tiene sentido en seres **animados** o **hablables**. Cuando el jugador manda hacer algo a otro personaje, usando una orden como «MANOLO, ABRE LA CAJA», el parser genera una llamada a la propiedad *ordenes* del objeto Manolo (siempre que Manolo tenga uno de los atributos antes mencionados). Esta rutina debe consultar la variable *accion* para saber

qué se le está mandando a Manolo (en este ejemplo tendría el valor `##Abrir`) y las variables ***uno*** para saber sobre qué objeto actuar (en este caso contendría el valor del objeto caja) y ***otro*** para el caso de que la orden llevara dos objetos (en nuestro ejemplo ***otro*** valdrá cero).

La rutina puede llevar a cabo la acción (lo cual suele exigir una programación compleja), o bien imprimir un mensaje a modo de respuesta, por ejemplo “Manolo dice: A mi no me des órdenes”.

Si el comando que se le da a Manolo tiene errores de interpretación (por ejemplo, se pide que abra una caja, cuando no hay tal caja a la vista, o peor aún se le pide un verbo no reconocido), entonces la variable ***accion*** toma el valor `##NoComprendido` y la variable ***tipoerror*** contiene un número de error (definido en la zona de constantes) que aclara un poco lo que ha pasado, para que Manolo pueda dar una respuesta coherente.

Si la rutina retorna ***true***, la librería entiende que Manolo se ha hecho cargo de la situación (ya sea porque ha llevado a cabo la orden, o porque ha escrito un texto indicando que no quiere) y la librería ya no intenta nada más. Si en cambio retorna ***false*** (o si la propiedad ***ordenes*** no existe en el objeto Manolo), la librería llamará a continuación a la rutina ***vida*** de Manolo. Si el parser había comprendido el comando que el jugador le mandaba a Manolo, la rutina ***vida*** recibirá una acción de tipo ***Orden***. De nuevo esta rutina puede consultar las variables ***accion***, ***uno*** y ***otro*** para decidir qué hacer. Si el parser no había entendido el comando, la rutina ***vida*** recibirá una acción tipo ***Responder*** y esta rutina deberá consultar las variables ***consultar_desde*** y ***consultar_num_palabras*** para averiguar lo que el jugador trata de decirle.

Si esta rutina devuelve ***false*** también, la librería emite el mensaje estándar: “Manolo tiene mejores cosas que hacer” (suponiendo que el parser haya comprendido la orden que le dabas a Manolo) o bien “No hay respuesta”, si no la había comprendido.

parse_nombre (Antes ***parse_name***) Esta propiedad es una rutina y puede tenerla cualquier objeto que no sea una habitación. Su cometido es ayudar al parser a saber si lo que ha escrito el jugador se refiere a este objeto. Si el objeto no tiene esta propiedad, el parser usará la rutina por defecto ***InterpretarNombre***, la cual comprueba si las palabras usadas por el jugador están en la lista de sinónimos o de adjetivos del objeto dado.

Si queremos cambiar este mecanismo de interpretación, podemos hacer que el objeto proporcione su propia función ***parse_nombre***. Desde dentro llamaremos repetidas veces a la función ***SiguientePalabra*** para obtener las palabras que el parser está tratando de descifrar. El cometido de esta rutina es decirle al parser cuántas de estas palabras (seguidas) son válidas para referirse a este objeto.

Por ejemplo, si el objeto se llama “botella de color rosa”, y el jugador escribe la orden «SACA LA BOTELLA DE COLOR ROSA DE LA CAJA», el parser, tras haber interpretado la palabra “SACA” como un verbo, llamará a todos los objetos que hay al alcance del jugador, para que le digan cuántas palabras a partir de la segunda

son aplicables a ese objeto. La rutina *parse_nombre* de la botella podría por ejemplo admitir todas las apariciones de “la” y de “de”, siempre que la palabra que va detrás de ellas sea “BOTELLA” o bien “COLOR” o bien “ROSA”. Si hiciera esto, su primera llamada a *SiguientePalabra* le retornaría ‘la’, por lo que de momento cuenta una palabra, la siguiente sería ‘botella’, por lo que ya tenemos dos palabras, a continuación viene ‘de’, que es guardado en la recámara a la espera de lo que haya detrás. La siguiente palabra es ‘color’, de modo que la admitimos y contamos también el ‘de’ que estaba en suspenso. Ya van cuatro palabras. *SiguientePalabra* nos devuelve ahora ‘rosa’, así que la contamos. Tenemos cinco. Las dos siguientes llamadas nos devuelven ‘de’ y ‘la’ y de momento las dejamos en reserva a la espera de la siguiente palabra. Finalmente obtenemos ‘caja’ que no es apropiada para este objeto, por lo cual no la contamos (y no contamos tampoco el ‘de’ y ‘la’ que teníamos en reserva). Resultado final: cinco palabras, lo cual es el valor que retornaría la rutina.

El parser llamará sucesivamente a los demás objetos y entenderá que el que más palabras encaje es el mejor candidato a lo que el jugador quiso decir.

En general, escribir rutinas de parsing es bastante complejo y no suele ser necesario, ya que la rutina por defecto *InterpretarNombre* hace bien su trabajo (y funciona con un algoritmo muy similar al descrito más arriba). Además la rutina *parse_nombre* puede ayudar al parser a determinar cuando dos objetos son indistinguibles (como una moneda de otra moneda idéntica), pero este uso de *parse_nombre* es más complejo y no se explica aquí.

plural (Antes *plural*) A veces aparecen en el juego muchos objetos idénticos (por ejemplo, muchas monedas). Si el jugador portara tres de estos objetos, al hacer inventario obtendría algo como “Llevas: una moneda, una moneda y una moneda”, lo cual es obviamente espantoso.

Para evitar esto, se da la propiedad *plural* a todas las monedas del juego, con el mismo valor para todas ellas (lo mejor sería definir una clase “moneda”, de la cual derivar cada moneda particular, así compartirán todo sin repetir el código). Cuando la librería se encuentre varios de estos objetos idénticos, los agrupará en uno solo usando el nombre que hayamos puesto en esta propiedad. Así el inventario diría ahora: “Llevas: tres monedas” (el cardinal “tres” lo calcula la librería, y la palabra “monedas” es la que aparece en la propiedad *plural* de estos objetos).

Para saber si dos objetos son idénticos o no, el parser llama a la función *Identicos* (véase), que forma parte también de la librería.

En caso de duda, el objeto puede proporcionar también la propiedad *parse_nombre* que es llamada en ciertos momentos (en una forma que no se explica aquí) para que decida si dos objetos son idénticos o no.

Cuando aparecen juntos varios de estos objetos idénticos, la librería emitirá un mensaje como “Puedes ver aquí seis monedas”. Cada moneda es un objeto separado, por lo que no se permite al jugador poner «EXAMINAR MONEDAS», sino que debería examinarlas de una en una. Lo que es peor, la palabra ‘monedas’ no es comprendida directamente, a menos que aparezca en la lista de la propiedad *nombre* de estos

objetos. Pero observar que 'monedas' no es un sinónimo válido de 'moneda' ya que uno se refiere a varios objetos y el otro a uno solo. Para este caso particular Inform admite una extraña sintaxis: puede ponerse en la propiedad *nombre* del objeto algo como 'monedas//p'. La doble barra seguida de la p indica a Inform que esta palabra es un plural para ese objeto. Así que si el jugador pone «COGE TODAS LAS MONEDAS» se convertirá en una acción *Coger* para cada moneda. También entiende cosas como «COGE TRES MONEDAS», o «COGE TODAS LAS MONEDAS EXCEPTO LA DE PLATA». Estas complejidades del parsing normalmente no son necesarias, pero hay que reconocer que quedan muy bien.

puerta_a (Antes *door_to*) En los objetos con el atributo **puerta**, esta propiedad indica a dónde lleva la puerta (es decir, en qué lugar aparecerá el jugador cuando la atraviese). Puede contener directamente el valor lugar, pero es más habitual que contenga una rutina que devuelva ese valor, de este modo pueden implementarse puertas bidireccionales, que lleven a lugares diferentes según desde qué lado se atraviesen.

Si la rutina devuelve cero (o si la puerta no proporciona esta propiedad) cuando el jugador intente pasar la librería le dirá que “no lleva a ninguna parte”. En realidad esto es más bien un error que no debería suceder.

reaccionar_antes (Antes *react_before*) Es una rutina que puede existir en cualquier objeto que no sea una habitación. Ante cualquier comando del jugador que haya sido interpretado con éxito, (por ejemplo «SALTA», o «CIERRA COFRE») el parser, antes que ninguna otra cosa, llama a la rutina *antes* de la habitación en la que el jugador está. Esta rutina puede impedir que el jugador lleve a cabo la acción, tal como se ha descrito previamente. Pero si no lo impide, a continuación la librería llamará a la rutina *reaccionar_antes* de todos y cada uno de los objetos que hay en esa habitación. Esta rutina tiene una estructura similar a la de la propiedad *antes*, es decir, una lista de acciones y qué hacer ante cada una de ellas. Es la oportunidad que tienen los objetos de impedir ciertas acciones cerca de ellos (o de causar efectos secundarios en esas acciones). Desde esta rutina pueden examinar los valores de las variables **uno** y **otro** (que contienen el objeto directo y el objeto extra que forman parte de la acción). Así, por ejemplo, puede hacerse que un objeto *espita de gas abierta* reaccione ante la acción «ENCENDER CERILLA», si ocurre en su proximidad.

Otro buen ejemplo puede ser la acción *Escuchar*. Normalmente, si el jugador pone «ESCUCHA» sin más aparece el mensaje “No escuchas nada raro”. Sin embargo, podemos hacer que el objeto “equipo Hi-Fi” intercepte esta acción y escriba en cambio “Oyes una música que no sabes de dónde viene” (si la variable **uno** vale cero, es decir, si el jugador no ha especificado objeto directo). Observar que ante el comando «ESCUCHA TIMBRE», en cambio, no sería adecuado que reaccionara el equipo musical (a menos que ponga algo del tipo “El equipo musical está tan alto que te impide escuchar otra cosa”). Si ningún objeto cercano reacciona (es decir si sus rutinas *reaccionar_antes* devuelven **false**), la librería seguirá su curso de acción normal (que en el caso de «ESCUCHA TIMBRE» sería activar la rutina *antes* del objeto timbre).

reaccionar_despues (Antes *react_after*) Análogo a *reaccionar_antes*, pero en esta ocasión la rutina es llamada una vez que la acción ha sido llevado a cabo (bien por el método *antes* de algún objeto, o por el método *vida* u *ordenes* de algún objeto animado, o bien por la propia librería).

salidas (*Nuevo*) Esta propiedad existe solamente si el juego define la constante **ADMITIR_COMANDO_SALIDAS** antes de incluir `EParser.h`. Esta rutina sirve para ayudar al comando «SALIDAS» a averiguar qué direcciones de la habitación son salidas evidentes (para mostrárselas al jugador en un mensaje como: “Salidas visibles: Norte, Este”).

Si esta propiedad existe, la librería la usará en la forma siguiente. Para cada punto cardinal, llamará a este método pasándole como parámetro una dirección (que puede ser **obj_n**, **obj_s**, etc). La rutina (si lo desea) imprimirá el nombre de esa salida y devolverá uno de los valores siguientes:

false Si deja que la librería decida si esa salida es evidente o no (la librería decidirá que sí es una salida si la correspondiente dirección en esa habitación tiene asignado algo que no sea una cadena de caracteres, es decir, lleva a otra habitación, a una puerta o a una rutina).

true Ya se ha escrito el nombre de esta salida (lo ha hecho la propia rutina *salidas*), la librería puede saltar a comprobar el siguiente punto cardinal.

2 No es una salida, la librería no debe mostrarla.

■ En cualquier otro caso, la librería entenderá que es una salida y la mostrará.

Normalmente esta propiedad no es necesaria, ya que la librería hace bien su trabajo de adivinar cuáles son las salidas de una habitación. Pero puede ser útil para habitaciones en las que una dirección no se convierte en una salida hasta que el usuario haya hecho algo concreto (derribar una pared, por ejemplo). En estos casos la rutina *salidas* comprobará si se cumple la condición necesaria y retornará 2 cuando quiera indicar que en esa dirección no hay salida y 3 (por ejemplo) cuando quiera indicar que sí la hay.

Esta propiedad y la acción *Salidas* son específicas de la librería española. En la librería original este comando podía lograrse incluyendo un módulo de ampliación, pero en esta versión forma parte ya de la librería estándar (si bien el comando sólo será permitido si el juego define la constante **ADMITIR_COMANDO_SALIDAS** como ya se ha dicho).

si_abierto (Antes *when_open*) Cuando la librería está mostrando los objetos que hay en una habitación (detrás de la descripción de la misma), si encuentra alguno **abrible**, y que además esté **abierto**, creará por defecto un mensaje del tipo “Puedes ver un cajón (en el que hay un papel)” o bien “Puedes ver un cajón (que está vacío)”. Es decir, que lista los contenidos del mismo.

Podemos cambiar ese comportamiento por defecto proporcionando la propiedad *si_abierto* (o su *alias si_abierta*). En este caso la librería no intenta crear descripción

alguna, sino que se limita a imprimir la que le proporcionemos aquí (en una línea separada).

A la hora de dar valor a esta propiedad hay que recordar que será impresa en una línea independiente, por lo que debe ser una frase completa, como por ejemplo “*Aquí hay un cajón abierto, que te invita a mirar dentro*”. Recordar asimismo que esta descripción sólo aparecerá si el objeto en cuestión está abierto. A menudo se usa con puertas, para mostrar una breve descripción de lo que se ve al otro lado. Esta propiedad también puede contener una rutina en lugar de una cadena. En este caso se asume que la rutina imprime la descripción correcta (la librería no imprime nada, si existe esta rutina, independientemente del valor que retorne ésta). Esto es útil de nuevo para el caso de la puerta, que puede tener descripción diferente según desde qué lado se mire.

Véase también *si_cerrado*.

si_apagado (Antes *when_off*) Cuando la librería está mostrando los objetos que hay en una habitación (detrás de la descripción de la misma), si encuentra alguno **conmutable**, y que además tenga el atributo **encendido** desactivado (por tanto está apagado) usará esta propiedad para generar una descripción de este objeto. Si la propiedad no existe lo añadirá simplemente a la lista de objetos “Puedes ver también...” sin indicar nada sobre su estado (encendido o apagado).

Podemos cambiar ese comportamiento por defecto proporcionando la propiedad *si_apagado* (o su *alias si_apagada*). En este caso el texto generado por esta propiedad se mostrará en una línea separada, antes de la lista “Puedes ver ...”. Así, por ejemplo podríamos crear un motor que en la propiedad *si_apagado* tenga la cadena Aquí hay un silencioso motor.

Véase también *si_encendido*.

si_cerrado (Antes *when_closed*) Cuando la librería está mostrando los objetos que hay en una habitación (detrás de la descripción de la misma), si encuentra alguno **abrible**, y que además no esté **abierto**, creará por defecto un mensaje del tipo “Puedes ver un cofre (que está cerrado)”.

Podemos cambiar ese comportamiento por defecto proporcionando la propiedad *si_cerrado* (o su *alias si_cerrada*). En este caso la librería no intenta crear descripción alguna, sino que se limita a imprimir la que le proporcionemos aquí (en una línea separada).

A la hora de dar valor a esta propiedad hay que recordar que será impresa en una línea independiente, por lo que debe ser una frase completa, como por ejemplo “*Aquí hay un cofre labrado con su pesada tapa bien cerrada*”. A menudo se usa con puertas, para mostrar una descripción del estilo “*Hacia el norte una puerta de aspecto sólido te bloquea el paso*”. Esta propiedad también puede contener una rutina en lugar de una cadena. En este caso se asume que la rutina imprime la descripción correcta (la librería no imprime nada, si existe esta rutina, independientemente del valor que retorne ésta). Esto es útil de nuevo para el caso de la puerta, que puede tener descripción diferente según desde qué lado se mire (por ejemplo, la puerta

antes mencionada, desde el otro lado debería decir “Hacia el sur ves una puerta de aspecto sólido”).

Véase también *si_abierto*.

si_encendido (Antes *when_on*) Cuando la librería está mostrando los objetos que hay en una habitación (detrás de la descripción de la misma), si encuentra alguno **conmutable**, y que además tenga el atributo **encendido** usará esta propiedad para generar una descripción de este objeto. Si la propiedad no existe lo añadirá simplemente a la lista de objetos “Puedes ver ...” sin indicar nada sobre su estado (encendido o apagado).

Podemos cambiar ese comportamiento por defecto proporcionando la propiedad *si_encendido* (o su *alias si_encendida*). En este caso el texto generado por esta propiedad se mostrará en una línea separada, antes de la lista “Puedes ver ...”. Así, por ejemplo podríamos crear un motor que en la propiedad *si_encendido* tenga la cadena “Aquí hay un motor tosiendo medio asfixiado”. La propiedad puede contener una cadena (como el ejemplo anterior) o una rutina que se ocupe de imprimir la cadena (útil cuando la descripción depende del estado de otras banderas).

Véase también *si_apagado*.

suma_al_alcance (Antes *add_to_scope*) Esta propiedad contiene una lista de objetos (separados por espacios) o bien una rutina.

La librería tiene complicados mecanismos para decidir cuándo un objeto está al alcance del jugador (al alcance significa en este caso que el jugador puede usarlo como objeto directo de una acción, por ejemplo «EXAMINA CASTILLO LEJANO», no significa que realmente esté al alcance de su mano. En el caso más sencillo, estarán al alcance todos los objetos que el jugador lleva consigo, los que están directamente en la habitación, y también (quizás) otros objetos que haya dentro de estos. Decimos quizás porque esto depende de si los objetos exteriores están abiertos o si son transparentes (si están cerrados y no son transparentes, los objetos que contengan no estarán al alcance). Cuando el jugador nombra un objeto que no está al alcance, la respuesta de la librería es “No veo eso que dices”.

Cuando un objeto que está al alcance tiene en la propiedad *suma_al_alcance* una lista de objetos, la librería automáticamente supondrá que todos ellos están también al alcance. Si la propiedad es una rutina, la librería la ejecutará. Desde dentro de la rutina se puede llamar a la función de librería *PonerAlAlcance* para poner dentro del alcance otros objetos.

No suele ser necesario usar esta propiedad. En el manual original el lector interesado puede encontrar algunos ejemplos.

tiempo_agotado (Antes *time_out*) Cada objeto lleva asociado un reloj que funciona marcha atrás. Este reloj se pone en marcha cuando se llama a la función de la librería *ArrancarReloj* (**objeto, cuenta**) y empieza con el valor “cuenta”. En cada turno se decrementa en 1. Cuando el reloj llega a cero, se ejecuta la rutina que está en esta propiedad.

El uso típico es el causar que las consecuencias de ciertas acciones del jugador no se manifiesten hasta un tiempo después. Por ejemplo, el hecho de «ROMPER VITRINA», puede activar el reloj de una alarma para que se dispare 10 turnos después. La rutina *tiempo_agotado* del objeto *alarma de ladrones* deberá cambiar el estado de la alarma a encendida (con **encendido**) y quizás mostrar un mensaje al jugador o activar alguna otra parte del juego.

Un reloj puede detenerse con PararReloj (**objeto**), entonces no se disparará su rutina *tiempo_agotado*. No se puede reanudar la cuenta del reloj donde se había dejado, pero puede volver a empezarse una cuenta nueva en cualquier valor dado.

tiempo_restante (Antes *time_left*) Es el número de turnos que le quedan a este objeto para que se dispare su rutina *tiempo_agotado*. El programa puede aumentar o disminuir el valor que hay aquí, o consultarlo para saber cuánto falta.

vida (Antes *life*) Esta propiedad es una rutina con una estructura idéntica a la de la propiedad *antes*. Su cometido es el mismo, es decir, interceptar ciertas acciones dirigidas a este objeto antes de que la librería intente llevarlas a cabo. La diferencia es que la rutina *vida* sólo es llamada para ciertas acciones que son habituales en la interacción con criaturas vivientes. En concreto, *Atacar*, *Besar*, *DespertarOtro*, *Lanzar*, *Dar*, *Mostrar*, *Preguntar*, *Responder*, *Hablar* y *Orden* (esta última es una acción falsa y la rutina *vida* debe consultar la variable **accion** para saber qué orden concreta se está pidiendo).

Si la rutina *vida* devuelve **true**, se entiende que ha procesado con éxito la acción y la librería no hace nada más. De lo contrario la librería ejecutará su método estándar, que consiste en llamar a la rutina *antes* de ese mismo objeto (donde hay otra oportunidad de capturar la acción). En realidad *vida* es un poco redundante con *antes*, pero esto ayuda a tener separadas las reacciones a los comandos para criaturas vivas (y simplifica la programación de clases genéricas).

Recordar que antes que a *vida*, se llama a *ordenes* si el jugador ha escrito algo del estilo «MANOLO, LO QUE SEA».

Capítulo 5

Acciones

Cuando el jugador escribe un comando como «PON LA BOLA EN LA MESA», esto se convertirá en una acción para el juego. La acción concreta que se producirá depende del tipo de objetos involucrados, por ejemplo, en este caso si la mesa es un **soporte** se generará la acción llamada *PonerSobre*, pero si en vez de una mesa fuese una caja (que en vez de **soporte** sería **recipiente**) la acción generada sería *Meter*.

La librería es capaz de generar hasta 100 acciones diferentes (más otras 19 para depuración, que no estarán disponibles para el usuario final, sino sólo para el programador). No obstante no hay que confundir los verbos que el juego comprende con las acciones que genera, ya que hay acciones que se disparan con diferentes verbos (por ejemplo, la acción *Examinar* se disparará cuando el jugador ponga «MIRAR OBJETO», «EXAMINAR OBJETO», «X OBJETO», «DESCRIBIR OBJETO», «M OBJETO»). Y a la inversa, un mismo verbo (por ejemplo «MIRAR») puede lugar a diferentes acciones, según el objeto sobre el que se aplique y las preposiciones que se usen, por ejemplo «MIRAR» a secas, produce la acción *Mirar* (que describe el lugar en que se halla el jugador), mientras que «MIRAR OBJETO» produce la acción *Examinar*, que describe ese objeto, y «MIRAR EN OBJETO» produce la acción *BuscarEn*, que normalmente describe los contenidos del objeto, etc.

En el fichero `Gramatica.h` es donde se definen todos los verbos que la librería comprende y qué acciones se generarán para cada posible verbo.

En esta sección daremos la lista de las acciones que la librería sabe manejar. Para cada posible acción existirá una rutina (cuyo nombre obligatoriamente termina en *Sub*, por lo que suelen ser llamadas rutinas VerboSub) encargada de ejecutar la acción por defecto correspondiente. Así, por ejemplo, la acción *Examinar* se lleva a cabo en la rutina llamada *ExaminarSub*. Esta rutina será llamada si la propiedad *antes* del objeto en cuestión decide no interceptar las acciones de *Examinar* sobre ese objeto.

El programador debe saber también que la variable global llamada *accion* contiene la acción solicitada por el jugador (la cual será una de las que se listan en esta sección), y que para tomar decisiones en base a esta variable, existen una serie de constantes predefinidas por el compilador cuyo nombre empieza por `##` seguido del nombre de la acción. Por ejemplo `##Examinar`. Estas constantes permiten comparaciones del tipo `if (accion == ##Examinar) ...`

Los nombres de todas las acciones están en infinitivo (aunque los verbos que el jugador teclea para dispararlas suelen ir en imperativo).

Las acciones de la librería se clasifican en cuatro tipos:

- **Meta-acciones.** Estas acciones no ocurren en el mundo del juego, sino en el computador en que se está jugando. Por ejemplo salvar y cargar partidas, abandonar el juego, etc. Cuando el jugador usa uno de estos comandos, el tiempo “no transcurre” (es decir, no se incrementa su contador de turnos).
- **Acciones que no hacen nada por defecto.** Son acciones que, aunque son permitidas por el parser, no tienen efecto sobre el juego sino que se limitan a imprimir algo del tipo “*No pasa nada especial*”. El programador debe usar la rutina *antes* de los objetos para que estas acciones hagan algo útil sobre algunos objetos concretos. Para los objetos que no capturen estas acciones, se aplicará el defecto. No sirve de nada poner una rutina *despues* para este tipo de acciones, ya que la librería nunca la llamará (recordemos que la rutina *despues* es llamada sólo después de que el comando en cuestión haya tenido éxito y la librería lo haya llevado a cabo, y que por definición los comandos de este grupo nunca tienen éxito).
- **Acciones que hacen algo.** Si la rutina *antes* del objeto no lo impide (retornando **true**), la librería hará algo que afectará al mundo del juego. La acción más típica de este grupo es «COGER», que causará que el objeto pase del lugar en que se halle, al inventario del jugador. Después de llevar a cabo la acción, y antes de imprimir un mensaje del tipo “*Hecho*”, la librería llamará a las rutinas *despues* de los objetos involucrados (a veces sólo para el primer objeto, y a veces para el **uno** y el **otro**, como ya se explicará). En esta rutina el objeto puede emitir algún mensaje adicional, o cambiar el estado de otros objetos del juego (por ejemplo, al abrir una caja fuerte y una vez que la librería se ha ocupado de todas las comprobaciones y que la caja realmente ha sido abierta, la rutina *despues* de la caja podrá activar una alarma).
- **Acciones falsas (*fake_actions*).** Estas acciones no son generadas por el jugador, sino por la propia librería. Por ejemplo, cuando el jugador pide «METE BOLA EN CAJA», se generará la acción *Meter* (que no es falsa), y el objeto “bola” será notificado de que el jugador intenta meterlo. Si la “bola” no pone inconvenientes, entonces la librería generará una acción falsa llamada *Recibir*, y se notificará a la “caja” de esta acción. Si la caja tampoco pone inconvenientes a recibir, la librería moverá la bola a la caja (suponiendo que se cumplen las demás restricciones, es decir, que la bola y la caja son visibles, que la caja es un **recipiente**, que está abierto, que no está ya demasiado lleno, etc...)
- **Acciones de depuración.** Estos verbos permiten hacer trampa en el juego y normalmente sólo estarán disponibles durante la fase de pruebas del juego. Cuando se haya superado esta fase, el programador desactivará todos estos verbos antes de distribuir el juego. Puede activarlos fácilmente mediante la constante **DEBUG**, o bien especificando el parámetro **-D** al compilador.

5.1. Meta-acciones

Nota: Para cualquiera de las meta-acciones, el verbo que el jugador puede usar está en infinitivo. En un principio la idea era distinguirlas de las acciones normales porque éstas últimas van en imperativo. Sin embargo, debido a que en las últimas versiones la librería admite también infinitivos para las acciones normales esta distinción ya es un poco obsoleta.

Observar también que muchos de los verbos admitidos están en inglés, pues a muchos jugadores les resulta más intuitivo poner «QUIT», «SAVE», «LOAD», que sus equivalentes en castellano.

ActivarNotificacion (Antes *NotifyOn*) Esta acción sólo estará presente en el juego si no se ha definido la constante **NO_PUNTUACION**. Lo que hace es activar la notificación de puntuación (el juego pondrá un mensaje cada vez que la puntuación cambie). Se genera mediante los verbos «NOTIFICAR SI», «NOTIFICAR ON», «NOTIFY ON» y «NOTIFY SI».

ActivarTranscripcion (Antes *ScriptOn*) Activa la transcripción a disco (se va copiando a un fichero todo lo que salga por pantalla). Se genera mediante los verbos «TRANSCRIPCION», «SCRIPT», «TRANSCRIPCION SI», «SCRIPT SI», «TRANSCRIPCION ON» y «SCRIPT ON».

DesactivarNotificacion (Antes *NotifyOff*) Esta acción sólo estará presente en el juego si no se ha definido la constante **NO_PUNTUACION**. Lo que hace es desactivar la notificación de puntuación (el juego ya no informará sobre los cambios en la puntuación). Se genera mediante los verbos «NOTIFICAR NO», «NOTIFY NO», «NOTIFICAR OFF» y «NOTIFY OFF».

DesactivarTranscripcion (Antes *ScriptOff*) Desactiva la transcripción a disco (deja de copiarse en fichero lo que sale por pantalla). Se genera mediante los verbos «TRANSCRIPCION NO», «TRANSCRIPCION OFF», «NOTRANSCRIPCION», «SCRIPT OFF», «SCRIPT NO», «NOSCRIPT» y «UNSCRIPT».

Dialecto (*Nuevo*) Muestra al jugador en qué dialecto se halla el juego. Se genera con el verbo «DIALECTO».

DialectoCast (*Nuevo*) Cambia el dialecto a “castellano”, con lo que el juego escribirá “coger” en lugar de “tomar”, y el verbo “coger” se interpretará por el parser para que genere la acción *Coger*. Se cambia a este dialecto con el comando «DIALECTO CASTELLANO».

DialectoSud (*Nuevo*) Cambia el dialecto a “sudamericano”, con lo que el juego dejará de escribir “coger”, escribiendo “tomar” en su lugar, y el verbo “coger” pasa a ser considerado como un taco por el parser (generará la acción *Tacos*). Se cambia a este dialecto con el verbo «DIALECTO SUDAMERICANO».

Finalizar (Antes *Quit*) Termina el juego. Es generada mediante los verbos «FIN», «ABANDONAR», «TERMINAR», «ACABAR», «QUIT», y «Q».

Lugares (Antes *Places*) Esta acción muestra una lista de todos los lugares en los que el jugador ha estado. En algunos juegos puede no existir (el programador elige si quiere permitir este verbo o no usando la constante **NO_LUGARES**). Se genera con los verbos «LUGARES» y «PLACES».

ModoM1 (Antes *LMode1*) Cambia el juego a modo *breve*. En este modo, al entrar en las habitaciones sólo se mostrará la descripción de la habitación si es la primera vez que se está en ella. De otro modo sólo se mostrará el “título” de la misma. Se genera mediante los verbos «BREVE», «NORMAL». Este es el modo por defecto, si el programador quiere usar otro modo por defecto en su juego, puede hacerlo modificando la variable *modomirar* en Inicializar.

ModoM2 (Antes *LMode2*) Cambia el juego a modo *largo*. En este modo siempre se muestran las descripciones de los lugares cuando se entra en ellos, aunque ya hayan sido visitados anteriormente. Se genera con los verbos «LARGO» y «VERBOSE».

ModoM3 (Antes *LMode3*) Cambia el juego a modo *superbreve*. En este modo nunca se muestran las descripciones de los lugares cuando se entra en ellos, sino tan sólo su título. Para obtener la descripción el jugador siempre puede «MIRAR» el lugar. Esta acción se genera con los verbos «SUPERBREVE» y «CORTO».

Objetos (Antes *Objects*) Esta acción muestra una lista de todos los objetos que el jugador ha tenido alguna vez en su poder, indicando entre paréntesis en qué lugar se halla cada uno de ellos en ese momento (o bien si han sido destruidos o han desaparecido). En algunos juegos puede no existir este verbo (el programador elige si quiere permitirlo o no usando la constante **NO_LUGARES**). Se genera con los verbos «OBJETOS» y «OBJECTS».

Pronombres (Antes *Pronouns*) Muestra una lista que especifica qué entiende el juego por “-lo”, “-la”, etc. El jugador puede usar los pronombres en comandos como «EXAMÍNALO». Normalmente “-lo” se referirá al último objeto masculino al que el jugador se haya referido, y “-la” al último femenino. Con este comando, no obstante, el jugador puede comprobar si los pronombres tienen los valores que él supone. Se genera con el verbo «PRONOMBRES».

Puntuacion (Antes *Score*) Esta acción sólo estará presente en el juego si no se ha definido la constante **NO_PUNTUACION**. Muestra la puntuación (resumida) del jugador. Se genera esta acción con los verbos «PUNTUACION» y «PUNTOS».

PuntuacionTotal (Antes *FullScore*) Esta acción sólo estará presente en el juego si no se ha definido la constante **NO_PUNTUACION**. Muestra la puntuación detallada y con explicaciones de las tareas cumplidas (si las hay en el juego). Se genera con los verbos «PUNTUACION DETALLADA», «PUNTUACION TOTAL», «TOTAL», «PT».

Reiniciar (Antes *Restart*) Vuelve a empezar el juego desde cero. Es activado por el verbo «REINICIAR».

Restaurar (Antes *Restore*) Carga una partida salvada desde el disco (se pedirá a continuación al jugador el nombre del fichero). Es activada por los verbos «CARGAR», «RECUPERAR», «RESTAURAR», «LOAD» y «RESTORE».

Salvar (Antes *Save*) Guarda en disco el estado de la partida actual. Esta acción es activada mediante los verbos «GUARDAR», «SALVAR» y «SAVE».

Verificar (Antes *Verify*) Comprueba si el juego está en buen estado (es una meta-acción bastante inútil, ya que generalmente el propio intérprete se ocupa de ello antes de dejarnos jugar). Se activa mediante el verbo «VERIFICAR».

5.2. Acciones que no hacen nada

Agitar (Antes *Wave*) Acción generada por los verbos «AGITA *cosa*», «ONDEA *cosa*» o «SACUDE *cosa*» (la *cosa* debe ser inanimada, pues de lo contrario se generará la acción *Atacar*). La respuesta estándar es “No lo tienes” si el objeto en cuestión no está en posesión del jugador, o bien “Te sientes ridículo agitando *la cosa*”.

Atacar (Antes *Attack*) Esta acción se genera ante cualquier intento de romper objetos o agredir a personajes. En concreto, los verbos «SACUDE *ser_vivo*», «SACUDE *ser_vivo*», «ROMPE *cosa*», «APLASTA *cosa*», «GOLPEA *cosa*», «DESTRUYE *cosa*», «PATEA *cosa*», «PISOTEA *cosa*», «ATACA *cosa*», «MATA *cosa*», «ASESINA *cosa*», «TORTURA *cosa*», «NOQUEA *cosa*», «LUCHA CON *ser_vivo*», «DA UNA PATADA A *cosa*», «DA UN PUÑETAZO A *cosa*», «DA UN GOLPE A *cosa*». Esta es una de las acciones que más sinónimos posee, puesto que además, en todos los casos anteriores *cosa* puede ser también un *ser_vivo*, en cuyo caso admite delante la preposición “a”, como por ejemplo «MATA A *ser_vivo*», «ATACA A *ser_vivo*», etc. Por defecto esta acción no causa ningún resultado, salvo el mensaje “La violencia no es la solución”. Quizás en algunos juegos pueda ser necesario distinguir entre “romper” y “atacar”, pero la librería estándar no hace tal distinción. La acción Romper no existe.

Atar (Antes *Tie*) Intentos de unir o enlazar (conectar) dos objetos. Se genera con los verbos «ATA *cosa*», «ATA A *ser_vivo*», «ATA A *ser_vivo* A *cosa*», «ATA *cosa* A *cosa*». En lugar de ATA puede usarse también ENLAZA (por ejemplo: «ENLAZA *cosa* A *cosa*»), UNE (por ejemplo: «UNE *cosa* A *cosa*»), ENCHUFA (por ejemplo: «ENCHUFA *cosa* A *cosa*»). Además también existen las formas «CONECTA *cosa* A *otra_cosa*» y «CONECTA *cosa* CON *otra_cosa*». (Sin embargo el verbo «CONECTA *cosa*» a secas genera la acción *Encender*). Sólo el primer objeto atado (si hay dos) recibirá la acción a través de su rutina *antes*. La respuesta estándar de la librería es “No lograrás nada así”.

Beber (Antes *Drink*) Se genera ante el verbo «BEBE *cosa*». Produce la respuesta por defecto “Eso no parece potable”.

Besar (Antes *Kiss*) Representa un acercamiento cariñoso a un ser animado. Se genera ante los verbos «BESA *ser_vivo*», «ABRAZA *ser_vivo*», «BESA A *ser_vivo*»,

«ABRAZA A *ser_vivo*». La acción es enviada a la rutina *vida* del ser, y si no hay resultado, a la rutina *antes* del mismo. Si tampoco hay resultado, la librería imprimirá el mensaje estándar “No creo que debas”.

Cantar (Antes *Sing*) Generada ante el verbo «CANTA». Por defecto la respuesta es “Cantas fatal”, pero puede capturarse la acción por la rutina *antes* de la habitación, o por *reaccionar_antes* de algún objeto cercano (quizás un oyente de los cantos).

Columpiar (Antes *Swing*) Generada ante los verbos «BALANCEATE EN *cosa*», «COLUMPIATE EN *cosa*», «MENEATE EN *cosa*». La respuesta estándar es “No es adecuado para columpiarse”. Un verbo curioso, en mi opinión.

Comprar (Antes *Buy*) Acción generada por los verbos «COMPRA *cosa*» y «ADQUIERE *cosa*». Por defecto produce la respuesta “No hay nada en venta”.

Consultar (Antes *Consult*) Esta acción se genera cuando el jugador trata de buscar información en libros o enciclopedias usando para ello los verbos «CONSULTA *cosa* SOBRE *palabras*», «CONSULTA *cosa* ACERCA DE *palabras*», «CONSULTA *palabras* EN *cosa*», «LEE SOBRE *palabras* EN *cosa*», «LEE *palabras* EN *cosa*», «BUSCA *palabras* EN *cosa*», «BUSCA EN *cosa* SOBRE *palabras*», «BUSCA EN *cosa* *palabras*», «BUSCA EN *cosa* ACERCA DE *palabras*». En todos estos casos el objeto *cosa* recibirá la acción *Consultar* (a través de la propiedad *antes*) y tendrá en la variable *otro* la primera palabra sobre la que el jugador desea encontrar información. Puede obtener el resto de palabras como se explica en la acción *Preguntar*. Observar que la *cosa* en cuestión puede ser un ser animado, pero la consulta no irá dirigida a su rutina *vida*, sino a la rutina *antes*. Por defecto produce la respuesta “No descubres nada interesante en *la cosa* sobre ese tema” (que no es una respuesta muy adecuada para objetos animados).

Cortar (Antes *Cut*) Se genera ante los verbos «CORTA *cosa*», «RASGA *cosa*», «CORTA *cosa* CON *objeto_poseido*» y «RASGA *cosa* CON *objeto_poseido*». Observar que el verbo “romper” no genera la acción *Cortar*, sino la acción *Atacar*. Por defecto produce la respuesta “Cortándolo no lograrás gran cosa”.

DespertarOtro (Antes *WakeOther*) Acción generada por los verbos «DESPIERTA *ser_vivo*» o «DESPIERTA A *ser_vivo*». En lugar de DESPIERTA puede usarse ESPABILA (por ejemplo: «ESPABILA A *ser_vivo*»). La respuesta estándar es “No parece necesario hacer eso”.

Despertarse (Antes *Wake*) Acción generada por los verbos «DESPIERTA», «ESPABILA», «DESPIERTATE», «ESPABILATE». La respuesta estándar es “La cruda realidad es que esto no es un sueño”.

Dormir (Antes *Sleep*) Generada ante los verbos «DUERME», «DESCANSA» o «RONCA». La respuesta estándar es “No estás especialmente somnoliento”.

Empujar (Antes *Push*) Acción generada sobre un objeto al empujarlo usando los verbos «EMPUJA *cosa*», «MUEVE *cosa*», «DESPLAZA *cosa*», «MENEA *cosa*», «EMPUJA A *ser_vivo*», «MUEVE A *ser_vivo*», «DESPLAZA A *ser_vivo*», «MENEA A *ser_vivo*» y

«PULSA *cosa*» (pensado para botones). Por defecto produce los mismos mensajes que la acción *Tirar*. Existe otra acción (*EmpujarDir*) que permite especificar en qué dirección empujar los objetos.

EmpujarDir (Antes *PushDir*) Esta acción tiene por objetivo mover grandes objetos (demasiado grandes para que el jugador pueda tomarlos) de un lugar a otro, empujándolos en la dirección de algún punto cardinal. El objeto tiene que manejar parte del movimiento desde su rutina *antes* (llamando a la función *PermitirEmpujarDir*), ya que si no lo hace no se podrá mover nada de todas formas. Se genera con los verbos «EMPUJA *cosa* HACIA *otra_cosa*», «MUEVE *cosa* HACIA *otra_cosa*» y «DESPLAZA *cosa* HACIA *otra_cosa*». En todos los casos se supone que la “otra cosa” ha de ser un punto cardinal. La respuesta estándar es “No creo que empujar la cosa sirva para nada”, pero también puede responder “Eso no es una dirección” o “No, no puedes en esa dirección”, dependiendo de lo que haga el objeto en su rutina *antes*.

Escuchar (Antes *Listen*) Acción generada por los verbos «ESCUCHA», «OYE», «ESCUCHA *cosa*», «OYE *cosa*» (la cosa puede ser un ser animado y se admite la preposición “a” delante, como en «ESCUCHA A *ser_vivo*»). Si hay objeto directo, éste recibirá en la rutina *antes* la acción. Si no hay objeto directo, puede usarse la rutina *antes* de la habitación para mostrar un mensaje distinto del estándar, que es “No escuchas nada fuera de lo común”.

Esperar (Antes *Wait*) Acción generada por el verbo «ESPERA» y su abreviatura «Z» (de “Zzzz”). Esta acción no hace nada, pero consume un turno. La librería escribe el mensaje estándar “Pasa el tiempo...”.

Excavar (Antes *Dig*) Se genera ante los verbos «CAVA EN *cosa*», «EXCAVA EN *cosa*», «CAVA *cosa*», «EXCAVA *cosa*», «CAVA *cosa* CON *objeto_poseido*», «EXCAVA *cosa* CON *objeto_poseido*», «CAVA EN *cosa* CON *objeto_poseido*» y «EXCAVA EN *cosa* CON *objeto_poseido*». Por defecto se genera el mensaje “Excavar no servirá de nada aquí”.

Fijar (Antes *Set*) Esta acción se produce ante los verbos «FIJA *cosa*» y «AJUSTA *cosa*». Ver también la acción *PonerA*. Por defecto produce la respuesta “Eso no puede regularse a ningún valor”.

Frotar (Antes *Rub*) Acción que recoge todos los intentos de frotado y limpieza, generada con los verbos «LAVA *cosa*», «LIMPIA *cosa*», «PULE *cosa*», «ABRILLANTA *cosa*», «FRIEGA *cosa*», «FROTA *cosa*». La *cosa* puede ser también un ser animado, en cuyo caso debe llevar la preposición “a” delante, como por ejemplo «LAVA A *ser_vivo*». La respuesta por defecto es “Ya está bastante limpio”.

Gesticular (Antes *WaveHands*) Acción generada por los verbos «AGITA LA MANO», «AGITA LAS MANOS», «SACUDE LA MANO», «SACUDE LAS MANOS» y «SALUDA CON LA MANO». La respuesta estándar es “Te ves ridículo gesticulando así”.

Girar (Antes *Turn*) Acción generada por los verbos «GIRA *cosa*», «ATORNILLA *cosa*» y «DESATORNILLA *cosa*». La respuesta estándar es idéntica a la de las acciones

Tirar y Empujar, y depende de los atributos del objeto girado. Si es **estatico**, “*Está firmemente sujeto*”, si es **escenario**, “*No eres capaz*”, si es un ser **animado**, “*Eso sería, como poco, maleducado*” y en el resto de los casos “*No ocurre nada, aparentemente*”.

Hablar (Antes *Tell*) Explicando cosas a otros personajes. Se genera ante los verbos «HABLA A *ser_vivo* DE *palabras*», «HABLA A *ser_vivo* SOBRE *palabras*», «HABLA CON *ser_vivo* SOBRE *palabras*», «HABLA CON *ser_vivo* DE *palabras*», «HABLA CON *ser_vivo* ACERCA DE *palabras*», «HABLA SOBRE *palabras* CON *ser_vivo*», «HABLA DE *palabras* CON *ser_vivo*», «HABLA DE *palabras* A *ser_vivo*», «HABLA *palabras* A *ser_vivo*», «HABLA *ser_vivo* DE *palabras*», «HABLA *ser_vivo* SOBRE *tema*», «HABLA *ser_vivo* *tema*». Por si fuera poco, todos los casos anteriores son comprendidos también si en lugar de HABLA se usa EXPLICA, NARRA o CUENTA. Por tanto el parser comprende cosas como «EXPLÍCALE EL PROBLEMA» (recordar que la partícula “-le” equivale al último ser animado al que nos hayamos referido).

Cualquiera que sea la forma que el jugador use, el parser dejará en la variable **uno** el objeto *ser_vivo* en cuestión, y en la variable **otro** la dirección de diccionario de la primera palabra en *palabras*. La rutina *vida* del *ser_vivo* debe descifrar esas palabras (como se explica en el epígrafe de la acción *Preguntar*). Si no lo hace, la librería generará la respuesta estándar “No has provocado ninguna reacción”, o bien, si no se encuentra al *ser_vivo* “Hablas solo durante un rato”.

IrAmbiguo (Antes *VagueGo*) Acción generada por los verbos «VE», «CAMINA», «VETE», «CORRE», «ANDA» y «VUELVE» cuando no se especifica la dirección en qué caminar. La librería se limita a imprimir el mensaje: “Tienes que especificar en qué dirección ir”.

Lanzar (Antes *ThrowAt*) Intentos de arrojar objetos contra otros objetos o personajes, usando los verbos «LANZA *objeto_poseido* A *cosa*», «LANZA *objeto_poseido* POR *cosa*», «LANZA *objeto_poseido* CONTRA *cosa*», «LANZA A *ser_vivo* POR *cosa*» y «LANZA A *ser_vivo* CONTRA *cosa*». Además en todos los casos anteriores puede usarse el verbo ARROJA en lugar de LANZA (por ejemplo: «ARROJA *cosa*» o «ARROJA *cosa* CONTRA *cosa*»). También existen las formas «TIRA *objeto_poseido* A *cosa*» y «TIRA *objeto_poseido* CONTRA *cosa*». En todos los casos el segundo objeto (contra el que se lanzan las cosas) puede ser un ser animado. El objeto que va a ser lanzado recibe la acción *Lanzar* en su rutina *antes* y si no impide el lanzamiento (retornando **true**), a continuación el objeto contra el que será lanzado (el **otro**) recibirá la acción-falsa *RecibirLanzamiento* en su rutina *antes*, donde puede consultar la variable **uno** para saber qué objeto ha sido lanzado y actuar en consecuencia. Si no lo impide (retornando **true**) la librería continuará con su acción estándar, que consiste en imprimir el mensaje “No serviría de nada” (si quien recibe el lanzamiento es inanimado) o “En el último momento te echas atrás” (si quien recibe el lanzamiento es un ser animado).

Llenar (Antes *Fill*) Se genera ante el verbo «LLENA *cosa*» o «RELLENA *cosa*». Observar que no se especifica con qué rellenarlo (se supone que el comando sólo se intentará

en presencia de líquidos libres, como ríos o mares). Por defecto produce la respuesta “No ves agua por ningún sitio”.

LoSiento (Antes *Sorry*) Generada ante cualquier intento de disculparse del jugador. Por ejemplo, ante cualquier frase que comience por «SIENTO», «LAMENTO», «PERDON», «SORRY», «PERDONA», «DISCULPA», «LO SIENTO» o «LO LAMENTO». La respuesta estándar de la librería es “Oh, no es necesario que te disculpes”.

MirarDebajo (Antes *LookUnder*) Acción generada por los verbos «MIRA BAJO cosa» y «MIRA DEBAJO DE cosa». La cosa es notificada a través de su rutina *antes*, y si no hace nada la librería mostrará el mensaje estándar “No ves nada interesante”.

Nadar (Antes *Swim*) Generada ante el verbo «NADA». La respuesta estándar es “No hay agua suficiente en la que nadar”. Conviene redefinir este mensaje en las habitaciones en las que sí haya bastante agua, lo cual puede hacerse en la rutina *antes* del lugar en cuestión.

No (Antes *No*) Acción generada cuando el jugador escribe «NX». Esto requiere una explicación más detallada. A veces el juego puede escribir preguntas como “¿Qué dices? ¿Estás loco?”. Algún jugador puede tomarse en serio esta pregunta y responder “sí” (o “no”). Por tanto estas respuestas deberían ser reconocidas por el parser, aunque sólo sea para responder a su vez algo como “Sólo era una pregunta retórica”. La forma de lograr esto es definir una acción *Si* que se genera cuando el jugador pone «SI», y una acción *No* que se genera cuando pone «NO». El problema es que “no” es a su vez una abreviatura para “noroeste”. La solución adoptada en esta traducción es que la palabra “no” será interpretada como “noroeste” siempre, excepto si la variable **PreguntaSiNo** está a 1 (en este caso, la palabra “no” es cambiada por “nx”, la cual da lugar a la acción *No*). Así pues, raramente se generará esta acción, a menos que el programador ponga explícitamente a 1 la variable **PreguntaSiNo**. Esta variable de todas formas vuelve a ponerse a cero automáticamente ante la siguiente respuesta del jugador, sea cual sea. La respuesta por defecto ante esta acción es “Sólo era una pregunta retórica”.

Cuando se pregunta al jugador cuestiones del tipo “¿Estás seguro?”, donde la respuesta del jugador sí importa, normalmente se llama a continuación a la función **SiONo**, la cual no tiene nada que ver con esta acción, sino que compara directamente la respuesta del jugador con las constantes **SI1__WD**, **SI2__WD**, **SI3__WD**, **NO1__WD**, **NO2__WD** y **NO3__WD** para saber si ha contestado “sí” o “no”.

Oler (Antes *Smell*) Generada ante los verbos «HUELE», «OLFATEA», «HUELE cosa», «OLFATEA cosa», «HUELE A ser_vivo», «OLFATEA A ser_vivo». En todos los casos se imprime el mensaje por defecto “No hueles nada extraño”, a menos que la rutina *antes* de la habitación o del objeto que está siendo olfateado haga otra cosa.

Pedir (Antes *AskFor*) Esta acción es generada por los verbos «PIDE A ser_vivoobjeto», «PIDELE A ser_vivoobjeto», «PIDE objetoAser_vivo» y «PIDELE objetoAser_vivo». En todos los casos **uno** contendrá el ser animado y **otro** el objeto solicitado. No obstante, el ser animado no recibirá esta acción, ya que el parser la transformará en

una orden “ser animado, da objeto a jugador”. Por tanto la rutina *ordenes* del ser animado debe contemplar el caso **Dar**, para interceptar esta acción. La respuesta por defecto es “El *ser_vivo* tiene mejores cosas que hacer”.

Pensar (Antes *Think*) Generada ante el verbo «PIENSA». Genera la respuesta estándar “Vaya. Qué buena idea”.

PonerA (Antes *SetTo*) Esta acción se genera cuando el jugador intenta manipular objetos que pueden ser ajustados a diferentes valores (por ejemplo, un dial de una emisora, la hora de un reloj). Para ello puede usar los verbos «PON *cosa* A *valor*» (como en «PON LA ALARMA A LAS 20:00»), «AJUSTA *cosa* A *valor*» «FIJA *cosa* A *valor*», «AJUSTA *cosa* EN *valor*» y «FIJA *cosa* EN *valor*». En todos los casos el objeto *cosa* recibirá en su función *antes* la notificación de esta acción, y podrá mirar la variable **otro** que contiene la primera palabra de lo que el jugador haya escrito como *valor*. También puede ser útil mirar a la variable **numero_interpretado** que contiene el número escrito por el jugador (caso de que haya escrito un número). Por defecto la respuesta es “Eso no puede regularse a ningún valor”.

Preguntar (Antes *Ask*) Es generada por los verbos «CONSULTA *ser_vivo* SOBRE *palabras*», «CONSULTA A *ser_vivo* SOBRE *palabras*», «CONSULTA SOBRE *palabras* A *ser_vivo*», «PREGUNTA *ser_vivo* SOBRE *palabras*», «PREGUNTA A *ser_vivo* SOBRE *palabras*», «PREGUNTA *ser_vivo* POR *palabras*», «PREGUNTA A *ser_vivo* POR *palabras*», «PREGUNTA *palabras* A *ser_vivo*», «PREGUNTA SOBRE *palabras* A *ser_vivo*», «PREGUNTA POR *palabras* A *ser_vivo*» y «PREGUNTA A *ser_vivo* ACERCA DE *palabras*». En todos estos casos, la variable **uno** contendrá el ser animado, y la variable **otro** la primera palabra de lo que se intenta preguntar. Al igual que en *Responder*, la rutina *vida* del ser animado deberá inicializar **np** con el valor de *consultar_desde* y realizar *consultar_num_palabras* llamadas a la función *SiguientePalabra*. Tras cada llamada recibirá una palabra más de la pregunta, y deberá actuar en consecuencia y retornar **true** para evitar que la librería escriba su mensaje estándar “No hay respuesta”.

Probar (Antes *Taste*) Intentos de saborear objetos, usando los verbos «SABOREA *cosa*», «PALADEA *cosa*», «LAME *cosa*», «PRUEBA *cosa*» y «CHUPA *cosa*». La *cosa* puede ser un ser animado, en cuyo caso requiere la preposición “a” delante («LAME A *ser_vivo*»). La respuesta estándar es “No saboreas nada inesperado”.

Quemar (Antes *Burn*) Acción generada por los verbos «ENCIENDE *cosa*», «PRENDE *cosa*» (siempre que la cosa en cuestión no sea **conmutable**, ya que si lo es se generará en cambio la acción *Encender*), «QUEMA *cosa*», «QUEMA A *ser_vivo*», «QUEMA *cosa* CON *objeto_poseido*», «QUEMA A *ser_vivo* CON *objeto_poseido*». Por defecto produce la respuesta “Con esa peligrosa acción no lograrías nada”.

Responder (Antes *Answer*) Es generada por los verbos «DI *palabras* A *ser_vivo*», «RESPONDE *palabras* A *ser_vivo*», «DILE *palabras* A *ser_vivo*», «RESPONDE A *ser_vivopalabras*», «DI A *ser_vivopalabras*», «DILE A *ser_vivopalabras*» y también si el jugador pone algo como “Manolo, texto” y el parser es incapaz de descifrar ese texto, se convertirá en una acción “di texto a Manolo”. En cualquier

caso, la variable **uno** contendrá la primera palabra del texto y la variable **otro** contendrá el ser animado a quien va dirigida la respuesta. Además, se llamará a la rutina *vida* de ese ser animado para que intente descifrar el resto del texto, el cual puede averiguar llamado repetidamente a la función *SiguientePalabra*, previa inicialización de la variable **np** con el valor *consultar_desde*. El número de palabras a leer lo indica la variable *consultar_num_palabras*. Por defecto la librería escribe el mensaje “No hay respuesta”.

Retorcer (Antes *Squeeze*) Generada ante los verbos «RETUERCE *cosa*», «APRIETA *cosa*», «ESTRUJA *cosa*», «TUERCE *cosa*». La *cosa* también puede ser un ser animado, en cuyo caso debe llevar delante la preposición “a” (por ejemplo: «APRIETA A *ser_vivo*»). La respuesta estándar es “¡Las manos quietas!” si se trataba de un ser animado, o bien “No consigues nada” si no.

Rezar (Antes *Pray*) Acción generada ante el verbo «REZA» u «ORA». No recibe objeto, pero la habitación puede interceptar esta acción en su rutina *antes*, de este modo en algunos lugares la oración puede dar resultado. Por defecto imprime el mensaje estándar “No obtienes nada práctico de tus oraciones”.

SaltarSobre (Antes *JumpOver*) Representa un salto por encima de un objeto. Se genera con los verbos «SALTA *cosa*», «BRINCA *cosa*», «SALTA SOBRE *cosa*», «BRINCA SOBRE *cosa*», «SALTA POR ENCIMA DE *cosa*» y «BRINCA POR ENCIMA DE *cosa*». Por defecto produce la respuesta “No lograrás nada así”.

Saltar (Antes *Jump*) Representa un salto en el sitio (para saltar por encima de objetos hay otra acción: *SaltarSobre*). Se genera ante los verbos «SALTA» y «BRINCA». Por defecto produce la respuesta “Saltas en el sitio, sin ningún resultado”. Puesto que no actúa sobre ningún objeto, la forma de interceptar esta acción es en la rutina *antes* de la habitación (de forma que en algunos lugares el saltar en el sitio puede dar lugar a otro mensaje o consecuencia).

Si (Antes *Yes*) Acción generada ante el verbo «SI». El verbo «SI» es comprendido independientemente del estado de la variable *PreguntaSiNo*. Ver la explicación de la acción *No* para más detalles.

Soplar (Antes *Blow*) Acción generada por el verbo «SOPLA *cosa*» Por defecto produce la respuesta “Tu soplido no produce ningún efecto”.

Soso (Antes *Mild*) Acción generada cuando el jugador emite algún insulto suave contra el juego, por ejemplo, con los verbos «ABURRIDO», «TONTO», «BOBO», «IDIOTA». Estos verbos pueden ir seguidos de cualquier texto adicional, por ejemplo «TONTO JUEGO DE LAS NARICES». Ante esta acción la librería emite el mensaje estándar “Bastante”.

Tacos (Antes *Strong*) Generada cuando la primera palabra del comando del jugador es un taco (por ejemplo «MIERDA», «JODER», etc.) La respuesta estándar es “Los verdaderos aventureros no usan ese vocabulario”. Nota: si el dialecto sudamericano está activo, el comando “coger” se considera un taco.

Tirar (Antes *Pull*) Acción generada sobre un objeto cuando se tira de él, usando el verbo «TIRA DE *cosa*». El mensaje impreso depende de los atributos del objeto. Si es **estatico**, dirá “Está firmemente sujeto”, si es **escenario** dirá “No eres capaz”, si es un ser **animado** dirá “Eso sería como poco maleducado”, y en otro tipo de objetos dirá “No ocurre nada, aparentemente”.

Tocar (Antes *Touch*) Generado por los verbos «TOCA *cosa*» y «TOCA A *ser_vivo*». También puede usarse, en lugar de TOCA el verbo PALPA. La respuesta estándar depende de cuál sea el objeto tocado; si es un ser animado “¡Las manos quietas!”, si es un ser inanimado “No notas nada extraño al tacto” y si es el propio jugador “Si crees que eso servirá de algo...”.

Tregar (Antes *Climb*) Acción generada por los verbos «TREPA A *cosa*», «ESCALA A *cosa*», «TREPA POR *cosa*», «ESCALA POR *cosa*», «TREPA *cosa*», «ESCALA *cosa*». Por defecto produce la respuesta “No creo que vayamos a lograr nada así”.

5.3. Acciones que pueden llegar a hacer algo

Abrir (Antes *Open*) Generada por los verbos «ABRE *cosa*» «ABRE A *ser_vivo*», «DESTAPA *cosa*» y «DESCUBRE *cosa*». Tras comprobar que el objeto es accesible al jugador, la librería chequeará si tiene el atributo **abrible** (si no lo tiene imprime “No es algo que pueda abrirse”) y si tiene **cerrojoechado** (imprimiendo entonces “Está cerrado con llave”) o si está **abierto** (imprimiendo “Ya estaba abierto”). Si pasa estas pruebas, la librería cambiará el valor del atributo **abierto**, llamará a la rutina *despues* del objeto en cuestión, y a menos que esta retorne **true**, escribirá un mensaje indicando al jugador el éxito de la acción. Este mensaje será simplemente “Abierto” en la mayoría de los casos, a menos que el objeto fuera un recipiente opaco, en cuyo caso será (por ejemplo) “Abres la caja descubriendo...” y seguidamente una lista de las cosas que tenía dentro¹.

Apagar (Antes *SwitchOff*) Se trata de apagar objetos conmutables (como un interruptor televisor, por ejemplo) pero no de apagar fuego (aunque siempre puede hacerse un objeto fuego que reaccione ante esta acción). Se genera con los verbos «APAGA *cosa*», «DESCONECTA *cosa*», «APAGA A *ser_vivo*» y «DESCONECTA A *ser_vivo*».

La librería se limita a comprobar que el objeto está al alcance del jugador, que sea **conmutable** (y si no imprime “No es algo que pueda apagarse”), que esté **encendido** (y si no imprime “Ya estaba apagado”), tras lo cual cambia el estado de su atributo **encendido** (desactivándolo) y llama a la rutina *despues* del objeto. A menos que esta retorne **true**, finalmente imprime “Apagas la cosa”.

Bajar (*Nuevo*)

¹ Si quisiéramos evitar este listado de contenidos, basta con hacer que la rutina *despues* del recipiente retorne **true**. Convendría también que en ese caso la rutina imprimiese “Abierto”, ya que de lo contrario el jugador no sería informado del éxito o fracaso de su acción.

BuscarEn (Antes *Search*) Esta acción puede entenderse bien como un intento de examen más a fondo, bien como un intento de conocer los objetos que hay dentro de otro (o encima de él). Se genera con los verbos «MIRA EN *cosa*», «MIRA DENTRO DE *cosa*», «MIRA SOBRE *cosa*», «MIRA A TRAVES DE *cosa*», «MIRA POR *cosa*» (como en “mira por la ventana”), «BUSCA EN *cosa*», «REGISTRA A *ser_vivo*», «REGISTRA *cosa*», «REGISTRA EN *cosa*» (en lugar de REGISTRA puede usarse REBUSCA).

La librería primero comprueba que no estemos a oscuras (mensaje: “Está muy oscuro”), y después que el objeto sea tocable. Si lo es, se mira si el objeto es **soporte** en cuyo caso se escribe algo como ‘Sobre *la cosa* puedes ver...’ seguido de la lista de objetos contenidos, o bien “No hay nada sobre *la cosa*”. Si es un **recipiente** se comprueba si está cerrado (en cuyo caso se escribe “No puedes ver lo que hay dentro de *la cosa* porque está cerrada”, y si está abierto se escribe algo como “En *la cosa* puedes ver...” seguido de la lista de objetos contenidos, o bien “*La cosa* está vacía”. Si el objeto no es ni **recipiente** ni **soporte** se escribe el mensaje “No encuentras nada interesante”.

Cerrar (Antes *Close*) Acción activada por los verbos «CIERRA *cosa*», «TAPA *cosa*», «CUBRE *cosa*». Ante esta acción la librería comprobará si el objeto en cuestión es **abrible**. Si no lo es imprimirá “No es algo que pueda cerrarse” y no hará nada. Si es **abrible**, comprobará si está **abierto** y si no lo está imprime el mensaje “Ya estaba cerrado” y no hará nada. Si pasa ambos test, imprimirá el mensaje “Cierras *la cosa*” y cambiará el atributo **abierto** de ese objeto, para reflejar que ya no está abierto.

Coger (Antes *Take*) Es la acción activada por los verbos «COGE *objetos*»², «TOMA *objetos*», «COGE A *ser_vivo*», «TOMA A *ser_vivo*», «QUITA *objetos_dentro* A *ser_vivo*». Observar que se pueden especificar varios objetos, o la palabra especial “todo”. En estos casos se generan una serie de acciones *Coger* separadas, una para cada objeto especificado.

Para esta acción la librería realiza las siguientes comprobaciones. Primero comprueba si el objeto a coger es el propio jugador (hay que comprobarlo todo), en cuyo caso emite el mensaje: “Siempre te tienes a ti mismo”, o si se trata de otro ser vivo, en cuyo caso emite el mensaje “No creo que *al ser_vivo* le gustara”³. Después se comprueba si el objeto y el jugador están en el mismo lugar (importante si el jugador está dentro de otro objeto), si lo que quiere coger no está también dentro, saldrá el mensaje “No está disponible”. Si el jugador está dentro del objeto que pretende coger (o sobre él), el mensaje será “Tienes que salirte *del recipiente*” (o bien “Tienes que bajarte *del soporte*”). Seguidamente se comprueba si el objeto está ya entre las posesiones del jugador, en cuyo caso se escribe “Ya tienes *el objeto*”. La siguiente comprobación es si se puede tocar el objeto (puede que esté dentro de un recipiente transparente, que sea parte de otro objeto o que esté en posesión de otro personaje) emitiendo el mensaje de error apropiado si el objeto no se puede coger.

² Sólo si está activo el dialecto castellano, ya que en el dialecto sudamericano COGE es transformado en JODE antes que entre el parser.

³ Si quisiéramos tener un ser vivo que el jugador pueda coger, debería ser la rutina *antes* del propio ser vivo la que lo permita y ejecute la instrucción `move self to jugador` para que dicho ser vivo pase a posesión del jugador.

Una vez se han pasado todos los tests anteriores, la librería se dispone a llevar a cabo la acción. Si el objeto está dentro de un recipiente (o sobre un soporte), la librería preguntará a dicho recipiente si quiere dejar salir al objeto, para lo cual ejecuta la rutina *antes* del recipiente usando la acción falsa *DejarSalir*. Si el recipiente retorna **true**, la librería no hace nada más (se supone que el recipiente ha escrito un mensaje explicando el problema). Si retorna **false** (o no tiene rutina *antes*) se continúa. Se mira finalmente si el objeto es **escenario** (imprimiendo el error “*Difícilmente podrías llevarte eso*”) o si es **estatico** (imprimiendo entonces “*Está fijado al sitio*”).

Finalmente, llegados a este punto el objeto puede cogerse. Solo queda por comprobar si el jugador lleva demasiadas cosas. De ser así, si además lleva consigo el **OBJETO_SACO**, automáticamente se moverá una de sus pertenencias a ese objeto (imprimiendo un mensaje que informa de ello), y si no lo lleva se imprimirá el mensaje “Ya llevas demasiadas cosas”. Una vez el jugador tiene sitio, se mueve por fin el objeto a sus posesiones y si el objeto estaba antes en un **recipiente** o **soporte**, se informa a este de que el objeto ha salido (llamando a su propiedad *despues* con la acción falsa *DejarSalir*). Finalmente la librería escribe el mensaje “Cogido” (o “Tomado” en el dialecto sudamericano).

Comer (Antes *Eat*) Acción generada por los verbos «COME *cosa_poseida*», «COMETE *cosa_poseida*», «TRAGA *cosa_poseida*», «INGIERE *cosa_poseida*», «MASTICA *cosa_poseida*». La librería comprueba que el objeto sea **comestible** en cuyo caso simplemente lo hace desaparecer y escribe el mensaje “Te comes *la cosa*. No está mal”. Si no era comestible imprimirá “Eso es simplemente incomedible” y no hará nada.

Es importante recordar que un objeto con el atributo **comestible** puede ser comido por el jugador, pero esto no tiene ningún efecto en el juego salvo la desaparición del objeto. Conviene capturar la acción *Comer* en la rutina *despues* de estos objetos si queremos que tengan algún efecto.

Dar (Antes *Give*) Dar una de las posesiones del jugador a otro ser vivo. Se genera con los verbos «DA *objeto_poseido* A *ser_vivo*», «DA A *ser_vivo* *objeto_poseido*», «DASELO A *ser_vivo*» (“-lo” se referirá al último objeto masculino referenciado), «DASELA A *ser_vivo*» (“-la” se referirá al último objeto femenino referenciado), «DALE *objeto_poseido* A *ser_vivo*», «DALE A *ser_vivo* *objeto_poseido*». En lugar de DA pueden usarse los sinónimos REGALA u OFRECE. La librería hará lo siguiente:

Primero comprueba que el objeto esté en poder del jugador, si no es así emite el mensaje “No tienes *el objeto*”⁴. Luego comprueba que el ser vivo al que se pretende dar el objeto no sea el propio jugador (hay que preverlo todo). Si fuera este el caso se imprime el mensaje “Manoseas *el objeto* durante un ratito, pero no consigues gran

⁴ Es muy raro que aparezca este mensaje, a menos que se redefina la gramática de este verbo, ya que en la librería la propia gramática exige que el objeto esté poseído para que el parser admita la orden. De hecho, si el jugador no tiene ese objeto, el parser generará automáticamente un “coger implícito”, tras el cual el jugador tendrá el objeto, o bien la acción habrá fracasado si no puede cogerlo. En cualquier caso el mensaje “No tienes *el objeto*” no aparecerá.

cosa”. Si el ser vivo no es el jugado, se llamará a la rutina *vida* de ese ser, para ver qué hace con el ofrecimiento. En esta rutina el programador habrá puesto código que examine la variable *uno*, que es el objeto ofrecido. Si esta función retorna **true** la librería no hará nada más. Si retorna **false** emitirá el mensaje estándar “*El ser_vivo no parece interesado*”, y el objeto seguirá en poder del jugador. Observar que la librería no mueve el objeto, si queremos que el ser vivo acepte el regalo, habrá que moverlo desde dentro de su rutina *vida*.

Dejar (Antes *Drop*) Acción generada por los verbos «DEJA *cosas_poseidas*», «DEJA A *ser_vivo*», o «TIRA *cosas_poseidas.O*» Observar que pueden especificarse varias cosas, e incluso es admitida la palabra “todo” para dejar todas las pertenencias. En caso de que se dejen varias cosas, la librería generará una acción *Dejar* separada para cada cosa. La habitación en que tiene lugar la acción, y después el objeto en cuestión serán avisados a través de la función *antes*, y pueden impedir que el objeto se deje retornando **true**. Si no lo impiden, la librería seguirá con su rutina estándar, la cual en primer lugar comprueba si el objeto a dejar es el propio jugador, en cuyo caso imprime el mensaje “Te falta destreza”, a continuación comprueba si el objeto a dejar estaba en la habitación, imprimiendo entonces “*La cosa ya estaba aquí*” seguidamente comprueba si el objeto está en posesión del jugador y si no imprime “No lo tienes”, a continuación mira si el objeto (que está en posesión del jugador) tiene el atributo puesto. En este caso genera una acción de **Desvestir**, tras la cual deja el objeto y escribe “Dejado”.

Desvestir (Antes *Disrobe*) Acción de quitarse cualquier objeto que se llevara puesto. Se activa por los verbos «QUITATE *objeto_poseido*» o «SACATE *objeto_poseido*». La librería comprobará si el objeto en cuestión tiene el atributo **puesto**, y si no lo tiene imprimirá el mensaje “No llevas puesto eso” y no hará nada, pero si lo tiene imprimirá el mensaje “Te quitas *la cosa*” y desactivará el atributo **puesto**.

EcharCerrojo (Antes *Lock*) Echa el cerrojo a un objeto (siempre que este objeto tenga el atributo **cerrojo**). Esta acción puede necesitar el especificar una llave o no. Se genera con los siguientes verbos (para los casos en que no es necesaria una llave): «PON CERROJO A *cosa*», «ECHA CERROJO A *cosa*», «COLOCA CERROJO A *cosa*» y también cambiando CERROJO por PESTILLO o CIERRE o con «CIERRA *cosa* CON PESTILLO». Por otro lado, para los casos en que el objeto necesita una llave se usará el verbo «CIERRA *cosa* CON *objeto_poseido*».

Ante esta acción la librería comprobará si el objeto está al alcance del jugador, como siempre, tras lo cual mirará si *cosa* tiene el atributo **cerrojo** (si no imprime “No parece tener ningún tipo de cerrojo”) o si ya tiene **cerrojoechado** (en cuyo caso imprime “*La cosa ya estaba cerrada con llave*” o “*La cosa ya tiene el cerrojo echado*”). Seguidamente se comprueba si el objeto que el jugador intenta usar como llave coincide con la propiedad *con_llave* de *cosa* si no coinciden, se emitirá el mensaje “No parece encajar en la cerradura” (o bien “Necesitas algún tipo de llave” si el jugador no ha especificado ninguna y el objeto exige una en su propiedad *con_llave*).

Si las pruebas anteriores son superadas, se cambia el valor del atributo **cerrojoechado** de *cosa* y se imprime el mensaje “Cierras *la cosa* con *el objeto_poseido*” (si se ha usado un objeto como llave), o bien “Echas el cerrojo a *la cosa*” (si no se ha usado llave).

Encender (Antes *SwitchOn*) Se trata de encender objetos conmutables (como un televisor por ejemplo). No se trata de encender fuegos, para esto está la acción *Quemar*. Se genera con los verbos «CONECTA *cosa*», «ENCIENDE *objeto_conmutable*» o «PRENDE *objeto_conmutable*»⁵.

La librería se limita a comprobar que el objeto está al alcance del jugador, que sea **conmutable** (y si no imprime “No es algo que pueda encenderse”), que no esté **encendido** (y si no imprime “Ya estaba encendido”), tras lo cual cambia el estado de su atributo **encendido** (activándolo) y llama a la rutina *despues* del objeto. A menos que esta retorne **true**, finalmente imprime “Enciendes *la cosa*”.

Entrar (Antes *Enter*) Se genera con el verbo «ENTRA» o «CRUZA», a secas (se supone que hay una entrada obvia en la descripción del lugar, como por ejemplo “Tienes una casa ante ti, con la puerta abierta”). La librería genera simplemente la acción *Ir* en la dirección “adentro” (la habitación donde esté el jugador debe proporcionar la propiedad *adentro* para indicar a dónde lleva este comando).

Examinar (Antes *Examine*) Una inspección cercana de un objeto o persona, que se logra con los verbos «MIRA A *ser_vivo*», «MIRA *cosa*», «MIRA HACIA *cosa*» (en lugar de MIRA puede usarse la abreviatura M), «EXAMINA *cosa*», «EXAMINA A *ser_vivo*» (en lugar de EXAMINA puede usarse la abreviatura X o EX, o los sinónimos INSPECCIONA, OBSERVA o DESCRIBE). También (curiosamente) el verbo «LEE *cosa*» causa la acción *Examinar*, pues no existe acción *Leer* en la librería (aunque no es difícil añadirla a un juego si lo requiriera). El cometido de esta acción es mostrar una descripción del objeto, para lo cual la librería dará los siguientes pasos:

Primero, se comprueba que el jugador no esté a oscuras (si así fuera se emitirá el mensaje estándar “Oscuridad, sustantivo: ausencia de luz que impide ver”). Si no está a oscuras, se intentará suministrar algo más de información sobre el objeto. Para ello se comprueba si tiene la propiedad *descripcion*, en cuyo caso se imprime o ejecuta esta propiedad, y seguidamente se añade la aclaración “*La cosa* está encendida” (o “apagada”) para los objetos que tengan el atributo **conmutable**. Si no tiene la propiedad *descripcion*, la librería comprobará si *cosa* es un recipiente, en cuyo caso mostrará sus contenidos generando la acción *BuscarEn*, y si no es recipiente mirará si es **conmutable**, en cuyo caso imprimirá si *cosa* está encendida o apagada. Si no es nada de lo anterior, emitirá el mensaje estándar “No observas nada especial en *la cosa*”.

Observar que para los objetos **conmutables** la librería siempre imprimirá su estado (encendido o apagado) cuando el jugador los examine. Si queremos impedir esto,

⁵ Si se aplican estos verbos sobre un objeto que no sea conmutable, se generará la acción *Quemar* en vez de *Encender*.

basta con quitar el atributo **conmutable**. A pesar de ello el objeto se puede *Encender* o *Apagar* igual, pero estas acciones debe manejarlas la rutina *antes* del objeto, pues la librería ya no lo haría al no ser el objeto **conmutable**.

Inv (Antes *Inv*) Muestra una lista de las posesiones del jugador, en el formato “horizontal” o “vertical”, dependiendo de cuál se haya usado por última vez (y por defecto al empezar el juego es el vertical). Se genera con los verbos «INVENTARIO», «I», «INV».

InvAlto (Antes *InvTall*) Muestra una lista de las posesiones del jugador, en formato “vertical” (una línea por cada objeto, y realizando indentaciones para los objetos que están dentro de otros). Se genera con los verbos «INVENTARIO BREVE», «I BREVE», «INV BREVE», «INVENTARIO ESTRECHO», «I ESTRECHO», «INV ESTRECHO».

InvAncho (Antes *InvWide*) Muestra una lista de las posesiones del jugador, en forma de frase “horizontal”. Se genera con los verbos «INVENTARIO ANCHO», «I ANCHO», «INV ANCHO», «INVENTARIO LARGO», «I LARGO», «INV LARGO».

Ir (Antes *Go*) Movimiento entre habitaciones. Se genera de forma implícita en algunos casos de la acción *Salir* (véase) y con los verbos «VETE *direccion*», «VETE HACIA *direccion*» y «VETE A *direccion*». En lugar de VETE pueden usarse los sinónimos VE, CAMINA, ANDA, CORRE, VUELVE. También puede usarse el verbo DIRECCION a secas (como «NORTE», «SUR», etc.) Este caso es especial, ya que las palabras “norte”, “sur”, etc. no son verbos, sino nombres de objetos (los objetos **obj_n**, **obj_s**, etc). Por ello este caso es manejado directamente por el parser quien compara la palabra recibida con los nombres de los objetos contenidos dentro del objeto **brujula**. Si encuentra una coincidencia, (pongamos que coincide con el nombre de **obj_n**), genera una acción *Ir*, usando como dirección el objeto hallado. Es decir, se convierte el comando «NORTE» en «VETE NORTE».

Una vez que la librería ha generado la acción *Ir*, siendo la variable **uno** el nombre de uno de los objetos contenidos en el objeto **brujula** la librería debe intentar el movimiento en esa dirección. El proceso es bastante complejo y se resume en los siguientes pasos:

- Primero, comprobar si el jugador está metido en (o encima de) otro objeto. Si fuera así, hay que preguntarle a ese objeto si se puede mover (podría ser un vehículo en el que el jugador viaja), para lo cual se llama a la rutina *antes* de ese objeto, la cual recibirá la acción *Ir*, y el objeto-dirección en la variable **uno**. Debe responder con uno de los siguientes valores:
 - 0 indica que el movimiento está prohibido (no es un vehículo después de todo) entonces la librería imprime “Tienes que bajarte *del soporte* antes” o “Tienes que salirte *del recipiente* antes”. Esto ocurrirá también si la rutina *antes* no existe o no contempla el caso *Ir*.
 - 1 Indica que se permite mover al objeto (es un vehículo). La librería moverá al objeto y al jugador, siempre que exista salida de la habitación en la dirección especificada.

- 2 Indica que el movimiento está prohibido (no es un vehículo) pero que no queremos que salga el mensaje del caso 0. (Quizás porque la propia rutina *antes* ya ha escrito un mensaje más apropiado). Por tanto la librería se asegurará de que jugador y objeto están en la misma habitación en que estaban, moviéndolos a ella, pero no hace nada más.
 - 3 El movimiento está permitido, pero ya se ha realizado desde la propia rutina *antes*. La librería no tiene que hacer nada más y termina.
- Una vez ha quedado claro que hay que mover al jugador (ya sea porque su “vehículo” ha respondido 1, o porque el jugador no estaba dentro de ningún objeto), la librería debe averiguar en qué dirección moverlo. La variable **uno** contiene un objeto de tipo **DireccionBrujula** (por ejemplo **obj_n**), y este objeto tiene una propiedad *puerta_a* cuyo valor es una propiedad de dirección (en nuestro ejemplo el valor sería *al_n*). Así pues, lo primero que hace la librería es obtener el valor de la propiedad *puerta_a* para el objeto-dirección.
 - Seguidamente comprueba si el valor obtenido es una propiedad de la habitación donde está el jugador (por ejemplo, si obtiene **al_n**, debe comprobar que la habitación tiene la propiedad **al_n**). Según lo que encuentre en esta propiedad, hará lo siguiente: Si la propiedad no existe o vale cero, usará la propiedad *no_puedes_ir* de la habitación para emitir un mensaje (que por defecto vale “No puedes ir por ahí”), y terminará. Si la propiedad (*al_n*) es una cadena de caracteres, la imprimirá y terminará (esta cadena se supone que es un mensaje del tipo “No puedo ir por el norte porque tal y tal”). Si es una rutina, la ejecutará y mirará lo que responde dicha rutina. Si responde **true** no hace nada más, si responde **false**, emitirá el mensaje contenido en la propiedad *no_puedes_ir* de esa habitación, y si retorna un objeto, intentará el viaje hasta ese objeto que llamaremos *objeto_destino* como se explica ahora.
 - Comprueba si *objeto_destino* es una puerta. En caso afirmativo, comprueba si tiene **oculto** (y entonces imprime “No puedes ir por ahí” y termina) y si tiene **abierto** (si no está abierta, emite el mensaje “No puedes pasar a través del objeto_destino”, o bien “No puedes trepar por *el objeto_destino*” o “No puedes bajar por *el objeto_destino*” dependiendo de en qué dirección se halle dicho objeto). Si la puerta está abierta, consulta su propiedad **puerta_a** para saber a dónde lleva. Si el resultado de la consulta es cero (o **false**), emite el mensaje “No puedes ir porque *el objeto_destino* no lleva a ningún sitio”, si es 1 no hace nada más (asumiendo que la rutina *puerta_a* del *objeto_destino* ya ha escrito algún mensaje indicando que no puede pasar). Si el resultado es un objeto, este será el nuevo *objeto_destino* (esta vez suele ser ya una nueva habitación).
 - Se transfiere al jugador el nuevo destino, (se lleva con él el vehículo si lo había), se mueven todos los objetos flotantes a la nueva localización, se actualiza la variable **localizacion** y como paso final se comprueba si el nuevo lugar está a oscuras. Si el nuevo lugar está a oscuras y también estaba a oscuras el lugar del que provenía el jugador, se llamará a la rutina CaminarAOscuras, la cual puede matar al jugador cambiando el valor de la variable **banderafin**.

Si esto no ocurre, la librería finalmente mostrará la descripción de la nueva habitación a la que hemos llegado.

Meter (Antes *Insert*) Intento de poner un objeto dentro de otro (si el otro es un **recipiente** o encima de él (si es un **soporte**). Se genera ante los verbos «PON *objetos* EN *recipiente*»⁶, «PON *objetos* DENTRO DE *cosa*», «PON A *ser_animado* EN *recipiente*», «PON A *ser_animado* DENTRO DE *cosa*». En lugar del verbo PON, puedes usar cualquiera de los sinónimos METE, ECHA, INSERTA o COLOCA. También existen las formas «DEJA *objetos* EN *recipiente*», «DEJA *objetos* DENTRO DE *recipiente*» y las mismas con SUELTA en lugar de DEJA. Además, los verbos «TIRA *objetos* EN *recipiente*» y «TIRA *objetos* DENTRO DE *recipiente*» también causan la acción *Meter*. Observar que pueden especificarse múltiples objetos (incluyendo la palabra “todo”) como objeto directo. En este caso el parser generará una acción *Meter* por cada uno de los objetos. Además se admite el verbo «TIRA *objeto_poseido* POR *cosa*» (como en “tira el papel por por el desagüe”). Observar que en cambio «TIRA *objeto* A *cosa*» provoca la acción *Lanzar*.

La librería lleva a cabo esta acción por defecto en la forma siguiente. Primero comprueba si el **recipiente** al que irán los objetos no contiene a su vez al jugador. En este caso convierte la acción *Meter* en *Dejar* esos objetos. También comprueba que el objeto realmente esté en posesión del jugador, y si no lo está escribe el mensaje estándar “Necesitas tener *el objeto* para poder meterlo donde sea” (si el objeto es inanimado) o bien “Antes tendrías que cogerlo, y no sé si se dejará” (si es animado). Seguidamente comprueba si el recipiente donde quieres meter el objeto está a su vez dentro de ese objeto. En este caso imprime el mensaje “No puedes meter un objeto dentro de sí mismo”. Seguidamente comprueba si el jugador puede “agarrar” el objeto en cuestión (puede que esté dentro de otra cosa cerrada, o que forme parte de otro objeto y sea inseparable, o que esté poseído por una criatura que no quiera soltarlo...) Si no pasa esta prueba se escribirá un mensaje de error adecuado (ver la acción *Coger* para más detalles) y se abandonará la acción para ese objeto.

Si pasa la prueba, se notifica a la función *antes* del recipiente que va a *Recibir* un objeto. Esta función puede imprimir un mensaje y retornar **true** si quiere impedirlo. Si lo desea puede examinar la variable *accion_recibir* que contendrá la acción que ha causado esta recepción (en este caso valdría *##Meter*, pero en otros casos podría valer *##PonerSobre*). Si no retorna **true**, la librería continúa indagando cosas. Lo siguiente que mira es que sea un recipiente abierto, y de no serlo emitirá el mensaje “*El recipiente* está cerrado”. Después mira si tiene realmente el atributo **recipiente**, y si no lo tiene escribirá “No se pueden meter cosas dentro *del otro_objeto*”. Seguidamente comprueba si el objeto que queremos meter lo llevamos puesto. De ser así genera automáticamente una acción *Desvestir*, para quitarse ese objeto antes de meterlo en otro lugar (e informa de ello con un mensaje).

La última comprobación que se hace es mirar si la *capacidad* del recipiente no está ya agotada (porque contenga otras cosas). En este caso se imprimiría el mensaje “No queda sitio en *el recipiente*”. Si pasa este último test, se mueve el objeto al

⁶ Nota importante, el comando «PON *objetos* EN *no_recipiente*» causa en cambio la acción *PonerSobre*.

interior del recipiente, se llama a las rutinas *despues* del objeto (con la acción *Meter*) y del recipiente (con la acción *Recibir*), y se imprime el mensaje estándar “Hecho”.

Meterse (Antes *GoIn*) Esta acción representa cualquier intento del jugador de moverse a otro objeto (ya sea para entrar en él, subirse a él, sentarse en él, pasar por él, etc.) Se genera mediante los verbos «VETE HACIA *cosa*», «VETE A *cosa*», «VETE POR *cosa*» o «VETE *cosa*» (y en lugar de VETE pueden usarse los verbos VE, CAMINA, ANDA, CORRE o VUELVE). También con los verbos «PASA POR *cosa*», «ENTRA EN *cosa*», «ENTRA POR *cosa*», «ENTRA A *cosa*», «ENTRA *cosa*», «CRUZA POR *cosa*», «CRUZA A *cosa*», «CRUZA *cosa*», «METETE EN *cosa*», «METETE POR *cosa*», «ATRAVIESA *cosa*», «SIENTATE EN *cosa*», «ECHATE EN *cosa*», «SIENTA EN *cosa*», «SAL POR *cosa*», «SALTE POR *cosa*», «BAJATE POR *cosa*», «BAJA POR *cosa*», «SUBETE A *cosa*», «SUBETE EN *cosa*», «SUBE A *cosa*», «SUBE EN *cosa*», «SUBE POR *cosa*», «SALTA A *cosa*» y «BRINCA A *cosa*».

Todas estas formas conducen a la misma acción *Meterse*. El programador debe tener cuidado al programar objetos como árboles de capturar los intentos de *Meterse* y de *Trepar*, observar que «SÚBETE AL ÁRBOL» causa la acción *Meterse*.

La librería hace un complicado trabajo con esta acción. En primer lugar, si *cosa* es una puerta o una dirección, se genera una acción *Ir* hacia ese objeto. Si el jugador ya estaba dentro, se genera el mensaje “Pero si ya estás en *la cosa*” (si era un **recipiente**) o bien “Pero si ya estás sobre *la cosa*” (si era un **sopORTE**). A continuación, se comprueba si el objeto es **entrable** y si no lo es se imprime el mensaje “No es algo donde puedas entrar” (o “No es algo donde puedas sentarte”, o “No es algo donde puedas subirte”, etc. dependiendo del verbo usado por el jugador). Si se trata de un recipiente cerrado, se imprime el mensaje “No puedes entrar en *la cosa* porque está cerrada”. Seguidamente se comprueba si *cosa* está sostenida por el jugador. De ser así se imprime el mensaje “Sólo puedes subirte en algo firmemente apoyado” o “Sólo puedes sentarte en algo firmemente apoyado” o similar. Seguidamente se mira si el jugador necesita salirse antes de algún otro objeto. Si es así, la librería genera automáticamente la acción *Salir*, informando de ello con un mensaje como “(te bajas del coche)”, para averiguar a continuación si necesita entrar en otros objetos antes de llegar al objeto pedido. Si es necesario, genera automáticamente las acciones *Meterse* necesarias, informando de ello con mensajes como “(te metes en el contenedor)”, hasta llegar finalmente a un punto desde el que pueda entrar al objeto *cosa*. Entonces mueve al jugador al interior de ese objeto e imprime el mensaje “Subes a *la cosa*” o “Te metes en *la cosa*” (según *cosa* sea un **sopORTE** o un **recipiente**).

Observar que en todos esos movimientos implícitos que hace la librería, podría abortarse la acción, si uno de los objetos intercepta la acción *Meterse* y no deja al jugador entrar, por eso se hacen de uno en uno en lugar de mover directamente al jugador del lugar en que se halla al lugar al que quiere ir.

Mirar (Antes *Look*) Acción generada al llegar a una habitación o cuando el jugador escribe «MIRAR» (a secas) o su abreviatura «M» (y, debido a un vicio del autor, se

mantiene también la forma «L» del inglés «LOOK»). El objetivo de esta acción es mostrar el título de la habitación, la descripción detallada de la misma (aunque esta puede ser omitida si el jugador ya ha estado antes en ese lugar), y la lista de objetos y personajes contenidos en esa habitación (salvo los que tienen el atributo **oculto** o **escenario**).

Para todo esto la librería sigue los siguientes pasos: primero comprueba que el jugador esté realmente en una habitación (aunque sea dentro en un objeto). Esto siempre será así, y si no lo fuera sería un error grave (“** Error de librería (10): el objeto-jugador está fuera del árbol de objetos”). Si todo es normal, la librería averigua lo que llama el “techo de visibilidad”, que es el objeto más externo al jugador que el jugador puede ver. Por ejemplo, imaginemos que el jugador se halla dentro de una caja transparente dentro de un armario cerrado, el cual se halla a su vez dentro de un camión que se halla en un aparcamiento. El “techo de visibilidad” del jugador es el armario, pues al estar cerrado ya no puede ver lo que hay fuera. Normalmente el techo de visibilidad del jugador será la habitación en que se halla, a menos que esté metido dentro de objetos como en el ejemplo anterior. Si el jugador está a oscuras, su techo de visibilidad será un objeto llamado **laoscuridad**, predefinido por la librería (la variable *localizacion* toma en este caso el valor **laoscuridad** y la variable *localizacion_real* contiene la habitación donde el jugador estaría si hubiese luz).

Una vez calculado el techo de visibilidad, si éste resulta ser la habitación donde se halla el jugador, (variable *localizacion*) se llamará a la rutina *inicial* de dicho lugar si el jugador acaba de llegar (si ya estaba en ese lugar, no es llamada). Esta rutina normalmente imprime un mensaje, justo delante de la descripción de la habitación (como por ejemplo “Tras una larga caminata llegas a...” pero puede usarse también para transferir al jugador a otro lugar si se cambia desde ella la variable *localizacion* (en este caso la librería rehace todos sus cálculos para hallar el nuevo techo de visibilidad una vez que la rutina *inicial* haya retornado). Si el jugador acaba de llegar a ese lugar, se llamará a la rutina *LugarNuevo* que por defecto está vacía (se trata de una función que el programador puede proporcionar, con código que se ejecutará cada vez que el jugador entre en un lugar. Por ejemplo podría ser un buen sitio para borrar la pantalla).

Lo siguiente es imprimir el “título” de la habitación, el cual es el nombre corto del techo de visibilidad antes hallado. Este saldrá en negrita (si se trataba de **laoscuridad**, su nombre corto se almacena en la constante **OSCURIDAD_TX** que por defecto vale “Oscuridad”). Si el jugador está dentro de un objeto, o subido en uno se especificará a continuación entre paréntesis junto al “título” de la habitación (en el ejemplo antes dado este título quedaría finalmente “Armario de caoba (en la caja de cristal)”, otro ejemplo podría ser “Dormitorio (sobre el taburete)”).

Seguidamente, si el techo de visibilidad coincide con la *localizacion* la librería construirá una descripción del lugar, llamando en primer lugar a la propiedad *describir* de dicho lugar, o si no existiera, usando la propiedad *descripcion* (si ésta tampoco estuviera definida, se daría el error de ejecución “** Error de librería (11): El lugar x no tiene propiedad descripcion”). No obstante, estas ejecuciones no sucederán cuando la descripción se deba a la llegada del jugador a la habitación si ya

había estado antes en ella (y si *modomirar* es distinto de 2, que es el modo “largo”).

Una vez descrito el lugar en que se halla el jugador, hay que describir los objetos que contiene. La función que muestra los contenidos de un lugar se llama *Local*. La librería, partiendo del objeto *techo* de visibilidad y hasta llegar al jugador, va llamando a la propiedad *descripcion_dentro* de cada objeto y seguidamente a *Local* para que muestre los contenidos de ese lugar. En la situación normal, en la que el jugador simplemente está en una habitación pero no dentro de otros objetos, todo esto se reducirá a llamar a *Local* una sola vez para que muestre los contenidos de la habitación. Véase la descripción de esta función para una descripción más detallada de qué objetos lista y cuáles no, y de cómo se usan las propiedades *inicial*, *describir*, *si_abierto*, *si_cerrado*, etc. para construir esta lista.

Cuando por fin todo ha sido escrito, la librería llamará a la función *RutinaMirar*, que por defecto está vacía, pero el programador puede usar para imprimir algo al final de cada habitación (por ejemplo una línea de guiones, como se hace en el juego “Vampiro”), y seguidamente a la rutina *PuntuacionLlegada*, la cual aumenta la puntuación del jugador en la cantidad **PUNTOS_LUGAR**, si la habitación no había sido visitada previamente y tenía el atributo **valepuntos**.

Mostrar (Antes *Show*) Generada por los verbos «MUESTRA *cosaA ser_vivo*», «MUESTRA A *ser_vivocosa*» y «MUESTRA *ser_vivocosa*» (esta última para que acepte comandos como «MUÉSTRALE EL DINERO», pues el verbo será roto en “muestra -le”). Además de MUESTRA se puede usar el sinónimo **ENSEÑA**.

La librería comprueba que el jugador tenga ese objeto en su poder (y si no imprime “No tienes *la cosa*”) y ejecuta la rutina *vida* del *ser_vivo*. Esta rutina puede consultar la variable **codigo_razon** que contiene la acción (##*Mostrar* en este caso) y la variable **uno** que contiene el objeto mostrado. En base a estos imprimirá un mensaje con la reacción del ser vivo y retornará **true**. Si no lo hace (o si retorna **false**), la librería mostrará el mensaje estándar “*El ser_vivo* no está impresionado”.

PonerSobre (Antes *PutOn*) Generada por los verbos «PON *cosas* EN *otra_cosa*» o «PON A *ser_vivo* EN *otra_cosa*» «DEJA *cosas* EN *otra_cosa*», «DEJA A *ser_vivo* EN *otra_cosa*», «TIRA *cosas* EN *otra_cosa*» (en todos estos casos, siempre que la *otra_cosa* no tenga el atributo **recipiente**, ya que en este caso la acción generada sería *Meter*), «PON *cosas* SOBRE *otra_cosa*», «PON A *ser_vivo* SOBRE *otra_cosa*», «PON *cosas* ENCIMA DE *cosa*», «PON A *ser_vivo* ENCIMA DE *otra_cosa*», «TIRA *cosas* SOBRE *otra_cosa*», «TIRA *cosas* ENCIMA DE *otra_cosa*». En todos los casos anteriores puede usarse en lugar de PON cualquiera de los verbos **METE**, **ECHA**, **INSERTA** o **COLOCA**, y en lugar de DEJA puede usarse **SUELTA**. Como se observa en muchos casos puede especificarse más de un objeto a poner, en cuyo caso la librería generará una acción *PonerSobre* para cada objeto por separado.

Para esta acción la librería realiza las siguientes comprobaciones: primero mira si la *otra_cosa* es el objeto **obj_abajo** (el suelo) o si el jugador está dentro dentro o sobre ella, en estos casos se generará la acción *Dejar*. A continuación comprueba si el objeto está en poder del jugador (si no lo está emite el mensaje “Necesitas tener *el objeto* para ponerlo donde sea” (si el objeto es inanimado) o “Antes tendrías

que cogerlo, y no sé si se dejará” (si es animado). Si el objeto que se quiere dejar contiene al objeto sobre el cual se quiere dejar (o si son el mismo objeto) se emitirá el mensaje “No puedes poner un objeto sobre sí mismo”.

A continuación se comprueba que el jugador pueda tocar el objeto *otra_cosa* sobre el que se quieren poner las cosas (no podría si estuviera dentro de un recipiente cerrado, si perteneciera a otro ser vivo o formara parte inseparable de otro objeto). Si puede, se consulta a ese objeto si está dispuesto a recibir. Para ello se llama a su rutina *antes*, con acción igual a *Recibir*, variable *uno* igual al objeto que va a ponerse sobre él y variable *accion_recibir* igual a *##PonerSobre*. Podemos entender la pregunta que hace la librería como “¿Te haces tú mismo cargo de este objeto?”; si la rutina responde *true*, la librería no hace nada más (normalmente se tratará del caso en que el objeto receptor rechaza el otro e imprime un mensaje como “No puedes poner tal cosa encima de tal otra”, por lo cual la librería no debe continuar). Si no responde *true*, la librería continúa su curso normal que consiste en comprobar si *otra_cosa* es un **soporte**⁷ y si no lo es se imprime “Poner cosas sobre la *otra_cosa* no servirá de nada”. Si ambos objetos están en posesión del jugador se imprimirá “Te falta destreza” (es decir, la librería sólo deja poner cosas sobre objetos que no estén en posesión del jugador, aunque esta regla puede cambiarse en la propiedad *antes* del objeto receptor).

Pasadas las pruebas anteriores, la librería inicia el movimiento del objeto. Primero comprueba si el objeto a mover tiene el atributo **puesto**, en cuyo caso genera una acción *Desvestir* para quitárselo antes. Seguidamente comprueba si el objeto cabe encima del otro (si no se supera el valor de su propiedad *capacidad*) y si no cupiera se emitirá el mensaje “No queda sitio en la *otra_cosa* para poner nada más”. De lo contrario, se mueve el objeto, se llama a las rutinas *despues* involucradas (a la del objeto movido con la acción *PonerSobre* y a la del objeto soporte con la acción *Recibir*) y finalmente se emite un mensaje que informe del éxito de la operación (el mensaje será “Hecho” para cada objeto movido si el jugador ha especificado varios, o bien “Colocas *el objeto* sobre la *otra_cosa*” si sólo había especificado uno).

QuitarCerrojo (Antes *Unlock*) Acción generada por los verbos «QUITA CERROJO A *cosa*», «QUITA PESTILLO A *cosa*», «QUITA CIERRE A *cosa*» (también puede usarse QUITALE en lugar de QUITA, y puede ponerse artículo delante de CERROJO, CIERRE o PESTILLO). En los casos anteriores el jugador no especifica llave alguna. Además pueden usarse los verbos «ABRE *cosa* CON *objeto_poseido*» si se quiere usar *objeto_poseido* como llave.

La librería, tras comprobar que *cosa* es accesible por el jugador, comprueba si tiene el atributo **cerrojo** (si no lo tiene imprime “No parece tener ningún tipo de cerrojo”), y si tiene **cerrojoechado** (si no, imprime “La *cosa* ya tenía abierto el cerrojo”). Seguidamente comprueba si el objeto especificado por el jugador como llave coincide con la propiedad *con_llave* de la *cosa*. Si no coinciden imprimirá “El *objeto_poseido* no parece encajar en la cerradura”, y si el jugador no ha especificado

⁷ Observar que se hace después de haber llamado a *antes*, por lo que es posible que un objeto pueda aceptar otros encima incluso sin ser declarado como **soporte**, basta con que su rutina *antes* se haga cargo de todo.

objeto alguno pero se requería uno, imprimirá “Necesitas algún tipo de llave”.

Si pasa las pruebas anteriores, la librería le quitará el atributo **cerrojoechado**, llamará a la rutina *despues* de cosa y escribirá un mensaje informando del éxito de la operación (puede ser “Abres *la cosa* con *el objeto_poseido*”, o bien “Quitás el cerrojo a *la cosa*” en caso de que no se requiriese llave alguna).

Sacar (Antes *Remove*) Extrae un objeto de otro. Sirve tanto para objetos que están *dentro* de otros, como para los que están *sobre* otros. Se genera con los verbos «COGE cosas DE *otra_cosa*», «TOMA cosas DE *otra_cosa*», «SACA cosas DE *otra_cosa*», «SACA A *ser_vivo* DE *otra_cosa*», «QUITA cosas DE *otra_cosa*», «QUITA A *ser_vivo* DE *otra_cosa*». Cuando el jugador especifique varias *cosas*, el parser generará una acción *Sacar* separada para cada objeto.

Lo primero que hace la librería es averiguar en qué objeto está el que se quiere sacar. Si este objeto es un recipiente cerrado, se emite un mensaje como “Por desgracia el bote está cerrado”. Observar que este recipiente en principio no tiene por qué coincidir con el objeto *otra_cosa* especificado con el jugador. Esto se comprueba a continuación y si no coinciden se emite el mensaje “¡Pero si no está ahí ahora!”. Si el recipiente en cuestión es un ser vivo se emite un mensaje como “Parece que pertenece al cerdo”.

A continuación la librería sigue exactamente igual que si se hubiera usado la acción *Coger*. Cuando termina y el objeto ha pasado finalmente a posesión del jugador, se informa de ello a la rutina *despues* del objeto movido (en esta ocasión la librería la llama dos veces, la primera con acción igual a *Coger* y la segunda con acción igual a *Sacar*), tras lo cual escribe el mensaje “Sacado”.

Salidas (*Nuevo*) Esta acción sólo estará disponible si el programador ha definido la constante **ADMITIR_COMANDO_SALIDAS** antes de incluir *Acciones.h*. En este caso la acción se genera ante los verbos «SALIDAS», «EXITS» o la abreviatura «X».

La librería debe ahora construir una lista de las salidas visibles de la habitación e imprimirla para el jugador. Para ello empieza por escribir “*Salidas visibles:*” y seguidamente recorre el objeto **brujula** y para cada dirección de la brújula averigua si la habitación tiene salida en esa dirección. Para averiguar esto primero intenta llamar a la propiedad *salidas* de la propia habitación, pasándole como parámetro la dirección en cuestión. Esta función (si existe) puede retornar diferentes valores indicándole a la librería qué ocurre con esa dirección en particular (véase la descripción de la propiedad *salidas*).

Si la propiedad *salidas* no existe para esa habitación, o retorna el valor **false** para esa dirección, la librería debe deducir por sí misma si hay salida o no en esa dirección. Para ello examina el valor de la propiedad correspondiente (por ejemplo, si está deduciendo la salida norte, examinará la propiedad *al_n* de la habitación). Si encuentra que no está definida, deducirá que no hay salida. Si encuentra que contiene una cadena, deducirá que no hay salida. Si encuentra que contiene una rutina, deducirá que no hay salida (al menos no es evidente, ya que la rutina puede dejar

pasar o no al jugador). Finalmente si encuentra que contiene una puerta o lleva a otra habitación deducirá que sí hay salida, e imprimirá el nombre de esa dirección.

Una vez haya comprobado todas las direcciones, comprobará cuántas salidas ha hallado. Si han resultado ser cero escribirá “Ninguna”.

Salir (Antes *Exit*) Salir del objeto en el que se halla el jugador, o bien abandonar la habitación a través de la salida llamada *afuera*. Esta acción se genera automáticamente tras una acción *Salirse*, o cuando el jugador usa uno de los siguientes verbos: «SAL», «FUERA», «AFUERA», «SALTE», «LEVANTATE», «BAJATE», «SAL FUERA», «SAL AFUERA». Ante este verbo la librería ejecuta la siguiente rutina estándar:

Primero comprueba si el jugador se halla en una habitación o dentro de un objeto. Si está en una habitación, y ésta define la propiedad *afuera*, usará esta propiedad generando la acción *Ir*, con destino el **obj_afuera**. Si está en una habitación pero ésta no define la propiedad *afuera*, se imprimirá el mensaje “No estás en ningún sitio del que debas salir” (o “bajarte”, o “levantarte”, según el verbo usado por el jugador). Si el jugador está dentro de un objeto **recipiente**, pero éste está cerrado, se imprimirá el mensaje estándar “No puedes salir *del objeto* porque está cerrado” de otro modo se mueve al jugador al objeto padre del objeto en que estaba metido (normalmente esto le sacará a la habitación, pero pudiera ser que estuviera a su vez dentro de otro objeto) y se le informa de ello con el mensaje “Sales *del objeto*” o “Bajas *del objeto*”, según el objeto sea **soporte** o **recipiente**. Tras este mensaje, se ofrece al jugador la descripción de la habitación, como si se hubiera dado el comando MIRA.

Salirse (Antes *GetOff*) Cuando el jugador está metido (o subido) en un objeto, esta acción causará que salga de él. Esta acción se genera con los verbos «SAL DE *cosa*», «FUERA DE *cosa*», «AFUERA DE *cosa*», «SALTE DE *cosa*», «BAJATE DE *cosa*», «LEVANTATE DE *cosa*» (para objetos como una cama) o «BAJA DE *cosa*». En todos estos casos la librería intentará sacar al jugador del objeto en que se halla metido (o subido) para lo cual simplemente comprobará que realmente el jugador está dentro de *cosa* (y si no estuviera imprimiría el mensaje estándar “Pero si no estás en *la cosa*”). Si efectivamente el jugador está dentro, la librería genera una acción *Salir*. La acción *Salirse* por tanto no es notificada al objeto *cosa* (aunque sí puede interceptarse en la rutina *antes* de la habitación donde se halla dicha *cosa*). Observar no obstante que el jugador puede salirse de cualquier objeto tecleando simplemente «SAL» (no es necesario especificar de dónde), y que en este caso no se genera la acción *Salirse*, sino *Salir*.

Subir (*Nuevo*) Generada por el verbos «SUBE», se convierte simplemente en una acción **Ir arriba**.

Transferir (Antes *Transfer*) Mueve un objeto que estaba dentro de un recipiente a otro recipiente. Se genera con los verbos «TRANSFIERE *cosa* A *otro_lugar*», o «CAMBIA *cosa* A *otro_lugar*»

La librería convierte esta acción en otras diferentes. En primer lugar intenta que el jugador coja *cosa* (haciendo todas las comprobaciones relacionadas con la acción

Coger) y si tiene éxito, entonces genera la acción *Meter* (si *otro_lugar* es **recipiente**) o *PonerSobre* (si *otro_lugar* es **soporte**) o simplemente *Dejar* (si no se aplica ninguno de los anteriores).

Vaciar (Antes *Empty*) Acción generada por el verbo «VACIA *cosa*». La librería simplemente transforma esta acción en la acción *VaciarEn*, siendo el **otro** objeto el suelo (es decir, el objeto **obj_abajo**).

VaciarEn (Antes *EmptyT*) Acción generada por los verbos «VACIA *cosa* DENTRO DE *otra_cosa*», «VACIA *cosa* EN *otra_cosa*», «VACIA *cosa* ENCIMA DE *otra_cosa*» o «VACIA *cosa* SOBRE *otra_cosa*». La acción por defecto de la librería pasa por las siguientes comprobaciones: primero comprueba que *cosa* sea **recipiente** (y si no lo es imprime “La *cosa* no puede tener cosas dentro”), seguidamente comprueba que esté **abierto** (si no lo está imprime “La *cosa* está cerrada”) después hace las mismas comprobaciones para *otra_cosa* (a menos que sea el suelo, en cuyo caso no las comprueba). Si *otra_cosa* no es **recipiente**, imprime “La *otra cosa* no puede tener cosas dentro”, y si no está **abierto** imprime “La *otra cosa* está cerrada”. Finalmente comprueba si hay algo dentro de *cosa* (si no hay nada imprime “La *cosa* ya está vacía”) y finalmente genera una serie de acciones **Transferir** usando como primer objeto cada uno de los objetos contenidos en *cosa* y como segundo objeto *otra_cosa*.

Vestir (Antes *Wear*) Generada por los verbos «PONTE *objeto_poseido*», «VISTE *objeto_poseido*», «VISTETE CON *objeto_poseido.O*» bservar que el objeto debe estar en posesión del jugador. La librería comprueba en primer lugar si el objeto está al alcance del jugador (puede estar metido dentro de otro objeto, aunque esté en sus posesiones), tras lo cual verifica que el objeto sea **prenda** (si no lo es imprime “¡No puedes ponerte eso!”), si está entre las pertenencias del jugador (si no lo está imprime “No lo tienes”), y si el jugador lo tiene **puesto** (en cuyo caso imprimiría “¡Ya lo llevas puesto!”).

Si supera las pruebas anteriores, simplemente se le da el atributo **puesto**, se llama a su rutina *despues* y se imprime un mensaje informando del éxito de la operación (“Te pones *el objeto_poseido*”)

5.4. Acciones de depuración

No se describen aquí. Véase el manual original.

ActivarAcciones (Antes *ActionsOn*)

ActivarAcentos (*Nuevo*)

ActivarComandos (Antes *CommandsOn*)

ActivarReloj (Antes *TimersOn*)

ActivarRutinas (Antes *RoutinesOn*)

ActivarTraza (Antes *TraceOn*)

Alcance (Antes *Scope*)

DesactivarAcciones (Antes *ActionsOff*)

DesactivarAcentos (*Nuevo*)

DesactivarComandos (Antes *CommandsOff*)

DesactivarReloj (Antes *TimersOff*)

DesactivarRutinas (Antes *RoutinesOff*)

DesactivarTraza (Antes *TraceOff*)

IrDonde (Antes *Gonear*)

LeerComandos (Antes *CommandsRead*)

MostrarObjeto (Antes *Showobj*)

MostrarVerbo (Antes *Showverb*)

NivelTraza (Antes *Tracelevel*)

Predecible (Antes *Predictable*)

XArbol (Antes *XTree*)

XIrA (Antes *Goto*)

XMover (Antes *XAbstract*)

XRobar (Antes *XPurloin*)

5.5. Acciones falsas

Estas acciones no pueden ser generadas por ningún verbo que escriba el jugador, sino que las genera la librería bajo ciertas condiciones.

DejarSalir (Antes *LetGo*) Es generada cuando el jugador intenta sacar algo de algún objeto. La librería llama a la rutina *antes* del recipiente especificando *DejarSalir* como acción. El recipiente puede devolver **true** para impedir que salga de él el objeto en cuestión.

ElMismo (Antes *TheSame*) Es generada por el parser cuando está tratando de deducir si dos objetos que aparentemente responden a un mismo nombre son en realidad idénticos (como dos monedas, por ejemplo). Normalmente el parser deduce esto comparando las propiedades *nombre* de los objetos, pero si los objetos en cuestión tienen una propiedad *parse_nombre*, entonces la librería no podrá deducirlo y cargará en la variable *accion* la acción falsa `##ElMismo`, y en las variables *parser_uno* y *parser_dos* los objetos que le plantean la duda, tras lo cual llamará a la rutina *parse_nombre* de ambos objetos. Si retornan -1 el parser los considerará indistinguibles, si retornan -2 los considerará diferentes entre sí. Otros valores indican a la librería que lo deduzca ella misma (mirando las propiedades *nombre* de los objetos).

HalladoPlural (Antes *PluralFound*) Esta acción falsa se usa en la rutina *parse_nombre* de un objeto como forma de indicar al parser que el nombre encajado es de hecho un plural que puede referirse a más de un objeto. Si la rutina *parse_nombre* detecta un plural de este tipo, asignará a la variable *accion_parser* el valor `##HalladoPlural` y retornará además el número de palabras encajadas, como es habitual.

ListaMiscelanea (Antes *ListMiscellany*) Esta es una acción falsa que se genera cuando la librería está creando una lista de objetos y quiere añadir al lado de cada uno una breve explicación entre paréntesis (indicando si el objeto está abierto, cerrado, encendido, apagado, etc.) Estos mensajes normalmente los crea la rutina *MLIdioma* pero si queremos sustituirlos por otros haciendo uso del objeto **MensajesLibreria**, la acción a capturar es *ListaMiscelanea*.

Miscelanea (Antes *Miscellany*) Esta acción es generada por la librería cuando va a mostrar un mensaje de error. Sirve para capturar esta acción en el objeto **MensajesLibreria** y así poder sustituir los mensajes por defecto. (Por ejemplo, el mensaje “¿Perdón?” que sale cuando el jugador no escribe nada, o “No conozco ese verbo” cuando la primera palabra del comando no es reconocida).

NoComprendido (Antes *NotUnderstood*) Esta acción se genera cuando el parser ha encontrado que el jugador ha escrito una orden destinada a otro personaje del juego (Por ejemplo “Manolo, mírate en el espejo”) pero en cambio no ha sido capaz de interpretar el comando que se le ha dado a ese personaje. En este caso el objeto Manolo recibe la orden *NoComprendido* en su rutina *ordenes* y puede intentar descifrarla por sí mismo o dar una respuesta conveniente. La variable *tipoerror* contiene una de las constantes definidas en la sección 9.5 de la página 100, lo cual puede ayudar a Manolo a dar una respuesta más adecuada al problema.

Orden (Antes *Order*) Cuando el parser llama a la rutina *ordenes* de un personaje para que lleve a cabo la acción solicitada por el jugador, si esta rutina retorna *false* (o si no existe), se le da otra oportunidad de reaccionar al personaje llamando a su rutina *vida*, pero en este caso la acción que recibirá esta rutina será la acción *Orden*. Para saber cuál fue exactamente la orden, esta rutina puede consultar las variables *accion*, *uno* y *otro*.

Prompt (Antes *Prompt*) Cuando llega el momento de imprimir el prompt, la librería usa el objeto **MensajesLibreria** para que lo haga. Este debe capturar la acción *Prompt* y retornar **true**. Si no lo hace la librería imprimirá su prompt por defecto, que es “>”.

RecibirLanzamiento (Antes *ThrownAt*) Tras una acción *Lanzar* que haya tenido éxito, la librería llamará a la rutina *antes* del objeto contra el cual se ha lanzado algo, pasándole como acción *RecibirLanzamiento*.

Recibir (Antes *Receive*) Tras una acción *Meter* o *PonerSobre* que haya tenido éxito, la librería llamará a la rutina *antes* del objeto dentro del cual (o sobre del cual) se ha puesto algo, pasándole como acción *Recibir*.

Capítulo 6

Variables

6.1. Variables que el programador debe conocer

accion (Antes *action*) Contiene la acción solicitada por el jugador. Esta variable puede compararse con las constantes que representan a cada acción (las cuales son el nombre de la acción precedido por ##). Por ejemplo: `if (accion == ##Meter) ...`

Normalmente no es necesario mirar a la variable **accion** porque la propia estructura de las propiedades *antes*, *despues* y *vida* es la de un `switch` cuya variable fuera **accion**. Sólo en algunos casos la variable que gobierna ese `switch` no tiene el mismo valor que la variable **accion** (en particular, cuando tienen lugar *acciones falsas*).

actor (Antes *actor*) Objeto (animado) que se supone que tiene que llevar a cabo la acción solicitada. Normalmente será el propio jugador (es decir, el objeto apuntado por la variable **jugador**, el cual suele ser el objeto llamado **objjugador** si bien en algunos juegos en los que el jugador pueda *encarnarse* en diferentes objetos podría cambiar). Cuando el usuario ha escrito algo del estilo «MANOLO, SALTA», la variable **actor** contendrá el objeto al cual se ha dirigido la orden.

banderafin (Antes *deadflag*) Esta variable indica si el juego ha terminado. La librería la consulta tras cada turno y puede tomar uno de los siguientes valores:

0 El juego continúa normalmente.

1 El juego ha terminado porque el jugador ha muerto (se imprime el mensaje estándar “*Has muerto*” y se da la oportunidad de reiniciar, cargar o salir).

2 El juego ha terminado porque el jugador ha ganado (se imprime el mensaje estándar “*Has ganado*” y se da la oportunidad de leer curiosidades sobre el juego).

Otros El juego ha terminado y la librería llama a la rutina `MensajeMuerte` para que imprima lo que ha pasado.

dialecto_sudamericano (*Nuevo*) Esta variable indica si el juego se halla en dialecto sudamericano (vale **true**) o no (vale **false**). El juego puede inicializar esta variable al valor preferido en la rutina Inicializar. Por defecto se inicializa con **false** pero el jugador puede cambiarla interactivamente haciendo uso del comando DIALECTO.

etapa_inventario (Antes *inventory_stage*) Cuando la librería está construyendo la lista de objetos visibles (o un inventario) lo hace en dos etapas para cada objeto de la lista. La primera etapa imprime el nombre del objeto y la segunda la información extra entre paréntesis del tipo “(encendido)”.

Para ello primero pone un 1 en esta variable, y llama a la rutina *listarse* del objeto (antes de haber escrito nada). Esta rutina puede devolver **true** y la librería ya no seguirá con él (por tanto evitará ser listado o puede cambiar el nombre con que aparecerá en los listados). Si devuelve **false** la librería continúa e imprime su nombre corto precedido del artículo indefinido.

Para la segunda etapa, la librería pone la variable **etapa_inventario** al valor 2 y llama de nuevo a la rutina *listarse* del objeto. Esta rutina puede elegir qué información adicional imprimir entre paréntesis (o no imprimir nada) y si retorna **true** la librería ya no seguirá. Si retorna **false** la librería imprimirá la información por defecto según los atributos del objeto. En particular, si se trata de un recipiente abierto la librería mostrará su contenido (para evitar esto podemos retornar **true** desde la rutina *listarse* como se ha dicho).

Esta misma variable es usada también cuando se tienen que inventariar objetos que deben aparecer juntos en el listado (porque proporcionan la propiedad *listar_juntos*). En este caso la librería pone un 1 en la variable **etapa_inventario** antes de llamar a la función *listar_juntos* del primero de los objetos (y este puede imprimir un mensaje de cabecera y retornar **false** para que la librería siga, o imprimir otra cosa y retornar **true** para que no siga). Tras ello, la librería imprimirá la lista de objetos que van juntos y una vez terminada pondrá un 2 en **etapa_inventario** y llamará de nuevo a *listar_juntos*, para que imprima algún mensaje final que cierre este grupo si lo desea.

jugador (Antes *player*) Esta variable representa al jugador. Inicialmente se tratará del objeto **objjugador**, pero puede cambiarse con la función CambiarJugador si el juego es de los que permite manejar a distintos personajes.

la_hora (Antes *the_time*) Mantiene la hora real.

localizacion (Antes *location*) Esta variable apunta al objeto habitación en el que se halla el jugador. Debe inicializarse en la función Inicializar al valor de la habitación en que queremos que comience el juego.

localizacion_real (Antes *real_location*) Mantiene el objeto habitación donde se halla el jugador cuando está a oscuras. Esta variable es necesaria porque cuando el jugador está a oscuras la librería lo mueve a una habitación ficticia llamada **laoscuridad**. Esta variable pues mantiene la localización en la que el jugador estaría si tuviese

luz. La librería necesita saberla para poder llevarle allí si enciende una luz, o para averiguar en qué direcciones puede moverse el jugador mientras está a oscuras (serían las salidas de la habitación apuntada por *localizacion_real*).

modo_notificar (Antes *notify_mode*) Vale 1 ó 0 según esté activada o no la notificación de puntuación. Puede cambiarse interactivamente con los verbos NOTIFICAR SI y NOTIFICAR NO. El programador puede darle un valor inicial en la rutina Inicializar (por defecto es 1).

no_interpretar (Antes *no_parse*) Esta variable está definida sólo si se ha definido la constante **TIEMPO_REAL** (véase) antes de incluir `EParser.h`. Esta variable normalmente vale 0, pero debe ser puesta a 1 desde la función `ActualizarTiempoReal` si se quiere ofrecer un nuevo prompt al jugador descartando lo que éste hubiera escrito hasta entonces (y además la función debe retornar *true*).

np (Antes *wn*) Esta variable se usa durante el parsing y puede ser necesaria si el programador pretende escribir una función *parse_nombre*. Contiene el número que hace la palabra “actual” dentro de la serie de palabras tecleadas por el jugador (la primera palabra tiene número 1, y la coma y el punto cuentan como otra palabra más). La función `SiguientePalabra` usa esta variable para retornar la siguiente palabra, por lo que el programador puede cargar un valor en ella antes de llamar a `SiguientePalabra` si quiere empezar el análisis desde otro punto.

otro (Antes *second*) En los comandos que llevan dos objetos (como por ejemplo «METE BOLA EN CAJA») esta variable apunta al objeto mencionado en segundo lugar. Si no hay segundo objeto la variable valdrá cero. Observar que un comando como «METE BOLA, CUBO Y TOALLA EN CAJA» sigue teniendo como valor de *otro* el objeto caja, ya que el parser descompone esta acción en tres equivalentes: «METE BOLA EN CAJA», «METE CUBO EN CAJA» y «METE TOALLA EN CAJA», que se ejecutan después una tras otra.

parser_listo (*Nuevo*) Esta variable sirve para controlar el modo de actuación del parser cuando recibe un comando que ha comprendido parcialmente pero, al final del mismo, encuentra palabras que no sabe interpretar. Por ejemplo, cuando tecleamos «RECOGER EL PAPEL CON CUIDADO» el parser comprende perfectamente el comando hasta que llega a «CON CUIDADO».

En estas situaciones, la librería actuará según el valor de la variable *parser_listo* (el cual vale por defecto 1, pero puede ser modificado por el programador en cualquier momento del juego). Los valores posibles de esta variable y su significado son:

- 0 La librería responderá simplemente “No entendí la última parte de la frase”, sin llevar a cabo ninguna acción.
- 1 La librería mostrará un mensaje como “No entiendo la última parte. ¿Quieres recoger el papel?”, mostrando al jugador lo que ha comprendido, y esperará que éste responda «SÍ» o «NO». Si recibe un sí como respuesta, intentará ejecutar el comando que ha comprendido (en nuestro ejemplo mostraría “Cogido”).

- 2 La acción que ha comprendido el parser se intentará llevar a cabo directamente, despreciando la parte no comprendida del comando introducido por el jugador y sin informarle a éste de la existencia de esta parte no comprendida.

puntuacion_tareas (Antes *task_scores*) Es un vector que debe definir el programador para asignar puntuaciones a los diferentes objetivos del juego. Véase la constante **HAY_TAREAS**.

puntuacion (Antes *score*) Contiene la puntuación del jugador.

PreguntaSiNo (*Nuevo*) Contiene un 1 si el juego acaba de hacer una pregunta retórica (es decir, una pregunta de la que en realidad no espera respuesta, como por ejemplo “¿Qué? ¿estás loco?”). El programador debe poner manualmente un 1 en esta variable si acaba de imprimir un mensaje de este tipo. La razón de ello es que el jugador podría intentar responder a esta pregunta con «NO» (que no está loco), pero esto sería comprendido por el juego como un intento de ir al noroeste. Sin embargo, si la variable **PreguntaSiNo** vale 1 y el siguiente comando es «NO», la rutina *EspanolAlInformes* cambiará el «NO» por «NX», lo que dará lugar a la acción *No*.

No confundir estas preguntas que tienen lugar dentro del juego con las que hace la función *SiONo*, que tienen lugar ‘fuera del juego’, ante comandos como «FIN». En este caso la respuesta del jugador no pasa por el parser, y no puede dar lugar a ninguna acción, sino que simplemente se analiza letra por letra para ver si coincide con las respuestas ‘sí’ o ‘no’.

tate_callao (Antes *keep_silent*) Si esta variable vale 1, la librería no imprimirá ningún mensaje mientras lleva a cabo las acciones. Su uso típico es cuando una acción dada debe llevar a cabo otra de forma implícita, por ejemplo, si queremos que al abrir con llave cierta puerta no sólo se cambie su atributo **cerrojoechoado**, sino que además cambie el atributo **abierto**, bastaría capturar la acción *QuitarCerrojo* en la rutina *despues* y hacer que genere la acción <Abrir puerta>. El único problema de esto es que la librería emitiría el mensaje “Abres la puerta” o similar. Si queremos evitarlo, pondremos un 1 en la variable **tate_callao** antes de generar dicha acción. Y hay que recordar volver a ponerla a cero una vez que la acción retorne.

turnos (Antes *turns*) Número de turnos consumidos por el jugador hasta el momento. Recordar que las meta-acciones no consumen turno, pero todas las demás sí. En particular conviene tener presente que el *Inventario* consume un turno.

uno (Antes *noun*) Esta variable apunta al objeto usado como objeto directo del verbo (si el verbo no lleva objeto directo valdrá cero). En los verbos que admiten múltiples objetos (como «SACA TODO DE LA CAJA»), el parser creará una lista con los objetos sobre los que debe actuar y generará una acción separada para cada uno de ellos, guardando en la variable **uno** el correspondiente objeto para cada acción generada.

6.2. Variables internas de la librería

accion_invertida (Antes *action_reversed*)

accion_parser (Antes *parser_action*)

accion_que_seria (Antes *action_to_be*)

accion_recibir (Antes *receive_action*)

amodo_noposeido (Antes *onotheld_mode*)

ancho_elemento (Antes *item_width*)

anidacion_menu (Antes *menu_nesting*)

aviso_avanzar (Antes *advance_warning*)

bandera_debug (Antes *debug_flag*)

bandera_demasiados (Antes *toomany_flag*)

bandera_deshacer (Antes *undo_flag*) Indica si el intérprete puede *deshacer* un turno.

bandera_guapo (Antes *pretty_flag*)

bandera_imprime_jugador (Antes *print_player_flag*)

bandera_paa (Antes *ats_flag*)

bandera_puesto_en (Antes *placed_in_flag*)

banderaluz (Antes *lightflag*)

banderamulti (Antes *multiflag*)

best_tipoerror (Antes *best_etype*)

buffer (Antes *buffer*)

buffer2 (Antes *buffer2*)

buffer3 (Antes *buffer3*)

bufferaux (*Nuevo*) Usada por la rutina *BuscarEnDiccionario*.

caso_nombre_corto (Antes *short_name_case*)

codigo_razon (Antes *reason_code*)

consultar_desde (Antes *consult_from*)

consultar_num_palabras (Antes *consult_words*)

contadorp2 (Antes *pcount2*)

contadorp (Antes *pcount*)

dat1 (Antes *inp1*)

dat2 (Antes *inp2*)

deducedesde (Antes *inferfrom*)

eeпа_desde (Antes *oops_from*)

eeпа_guardado (Antes *saved_oops*)

elemento_menu (Antes *menu_item*)

elsiguiente (Antes *lookahead*)

encajado_desde (Antes *match_from*)

error_alcance (Antes *scope_error*)

estadio_alcance (Antes *scope_stage*)

estilo_ac (Antes *c_style*)

estilo_inventario (Antes *inventory_style*)

filtro_token (Antes *token_filter*)

gramatica_normal_tras (Antes *usual_grammar_after*)

herobj (Antes *herobj*)

himobj (Antes *himobj*)

hora_freq (Antes *time_rate*)

hora_incr (Antes *time_step*)

indef_casos (Antes *indef_cases*)

indef_esperado (Antes *indef_wanted*)

indef_numero_en (Antes *indef_nspec_at*)

indef_posibambig (Antes *indef_possambig*)

indef_propietario (Antes *indef_owner*)

indef_suponer_p (Antes *indef_guess_p*)

indef_tipo (Antes *indef_type*)

indentacion_eld (Antes *wlf_indent*)

interprete_estandar (Antes *standard_interpreter*) Contiene el número de versión del estándar de Máquina Z que el intérprete dice que soporta.

itobj (Antes *itobj*)

just_undone (Antes *just_undone*) Se utiliza para evitar que se puedan deshacer dos turnos consecutivos.

lastdesc (Antes *lastdesc*)

lineaEstado1 (Antes *sline1*)

lineaEstado2 (Antes *sline2*)

listando_junto (Antes *listing_together*)

localizacion_actor (Antes *actors_location*)

long_encajado (Antes *match_length*)

mant_np (Antes *hb_wn*)

meta (Antes *meta*)

ml_n (Antes *lm_n*)

ml_o (Antes *lm_o*)

modo_indef (Antes *indef_mode*)

modo_mantenido (Antes *held_back_mode*)

modo_multi (Antes *multi_mode*)

modo_noposeido (Antes *notheld_mode*)

modo_transcripcion (Antes *transcript_mode*)

modomirar (Antes *lookmode*)

multi_contexto (Antes *multi_context*)

multi_esperado (Antes *multi_wanted*)

multi_hallado (Antes *multi_had*)

nextbest_tipoerror (Antes *nextbest_etype*)

no_deducir (Antes *dont_infer*)

no_poseido (Antes *not_holding*)

nombre_elemento (Antes *item_name*)

nsns (Antes *nsns*)

num_palabras (Antes *num_words*)

numero_de_clases (Antes *number_of_classes*)

numero_de_encajados (Antes *number_matched*)

numero_especial1 (Antes *special_number1*)

numero_especial2 (Antes *special_number2*)

numero_especial (Antes *special_number*)

numero_interpretado (Antes *parsed_number*)

objeto_pronombre (Antes *pronoun_obj*)

objeto_raiz (Antes *top_object*)

old_herobj (Antes *old_herobj*)

old_himobj (Antes *old_himobj*)

old_itobj (Antes *old_itobj*)

paa_tfl (Antes *ats_hls*)

palabra_especial (Antes *special_word*)

palabra_pronombre (Antes *pronoun_word*)

palabra_verbonum (Antes *verb_wordnum*)

palabra_verbo (Antes *verb_word*)

parametros_deseados (Antes *params_wanted*)

parametros (Antes *parameters*)

parse (Antes *parse*)

parse2 (Antes *parse2*)

parseaux (*Nuevo*)

parser_dos (Antes *parser_two*)

parser_inflexion (Antes *parser_inflection*)

parser_trace (Antes *parser_trace*)

parser_uno (Antes *parser_one*)

permitir_plurales (Antes *allow_plurals*)

PreguntaCualExactamente (*Nuevo*)

prepdeducida (Antes *inferword*)

punt_anterior (Antes *last_score*) Almacena la puntuación del jugador en el turno anterior, para saber si ésta ha cambiado.

puntos_cosas (Antes *things_score*)

puntos_sitios (Antes *places_score*)

quitacentos (*Nuevo*)

razon_alcance (Antes *scope_reason*)

regla_coger_todo (Antes *take_all_rule*)

relojes_activos (Antes *active_timers*)

tamano_listando (Antes *listing_size*)

tdatos_encontrado (Antes *found_tdata*)

techo_de_visibilidad (Antes *visibility_ceiling*)

tipoerror (Antes *etype*)

token_alcance (Antes *scope_token*)

ttipo_encontrado (Antes *found_ttype*)

valor_lj (Antes *lt_value*)

x_cuenta_ambito (Antes *x_scope_count*)

xcommsdir (Antes *xcommsdir*)

Capítulo 7

Objetos

brujula (Antes *compass*) Este objeto debe contener dentro otra serie de objetos, cuyos nombres serán interpretados por el parser como nombres de direcciones. Por defecto la librería define un conjunto de objetos y los pone dentro de la brújula para las direcciones habituales. Estos objetos son **obj_n**, **obj_s**, **obj_e**, **obj_o**, **obj_ne**, **obj_no**, **obj_se**, **obj_so**, **obj_arriba**, **obj_abajo**, **obj_adentro** y **obj_afuera**. Excepto los dos últimos, que son obligatorios en todo juego Inform, los demás pueden eliminarse de la brújula si el programador define la constante **SIN_DIRECCIONES** antes de incluir `EParse.h`. En este caso se espera que el programador defina sus propios objetos y los ponga dentro de la brújula. Cada uno de estos objetos, además de su nombre, tendrán la propiedad *direcc_puerta* y esta propiedad ha de tomar uno de los valores *al_n*, *al_s*, *al_e*, *al_o*, *al_ne*, *al_no*, *al_se*, *al_so*, *arriba*, *abajo*, *adentro* y *afuera* indicando en qué dirección se moverá el jugador si teclea el nombre de ese objeto.

laoscuridad (Antes *thedark*) Este objeto es una habitación ficticia a cuyo interior será movido automáticamente el jugador siempre que se halle a oscuras. El programador puede comprobar si el jugador está a oscuras comparando la variable *localizacion* con este objeto. Si son iguales (`if (localizacion == laoscuridad) ...`) el jugador está a oscuras. En ese caso puede consultar la variable *localizacion_real* para saber cuál es la habitación en que *realmente* está el jugador (es decir, la que sería si hubiera luz).

LibreriaInform (Antes *InformLibrary*) Este objeto encapsula el motor del juego. El programador no necesita conocerlo ni usarlo. Baste decir que la función *Main* de Inform (que es la primera en ejecutarse siempre) se limita a llamar a la propiedad llamada *jugar* de este objeto.

MensajesLibreria (Antes *LibraryMessages*) Este objeto no está definido por la librería estándar, sino que queda a la libertad del programador definir uno. Si lo hace, la librería llamará a la rutina *antes* de este objeto cada vez que deba imprimir uno de los mensajes estándares. En dicha rutina se puede capturar la acción para cambiar el mensaje por defecto. El programador deberá consultar además las variables *ml_n* para saber el número de mensaje a imprimir para esa acción y *ml_o* para saber el

objeto sobre el que se ha intentado la acción. Si la rutina *antes* escribe algo, debe retornar **true** para evitar el mensaje estándar. Si retorna **false** la librería imprimirá su mensaje estándar de todas formas.

obj_abajo (Antes *d_obj*) Véase el objeto **brujula**.

obj_afuera (Antes *out_obj*) Véase el objeto **brujula**.

obj_arriba (Antes *u_obj*) Véase el objeto **brujula**.

obj_dentro (Antes *in_obj*) Véase el objeto **brujula**.

obj_e (Antes *e_obj*) Véase el objeto **brujula**.

obj_ne (Antes *ne_obj*) Véase el objeto **brujula**.

obj_no (Antes *nw_obj*) Véase el objeto **brujula**.

obj_n (Antes *n_obj*) Véase el objeto **brujula**.

obj_o (Antes *w_obj*) Véase el objeto **brujula**.

obj_se (Antes *se_obj*) Véase el objeto **brujula**.

obj_so (Antes *sw_obj*) Véase el objeto **brujula**.

obj_s (Antes *s_obj*) Véase el objeto **brujula**.

objjugador (Antes *selfobj*) Objeto que normalmente será el jugador (a menos que se haya dado otro valor a la variable **jugador** o se haya llamado a **CambiarJugador**).

ParserInform (Antes *InformParser*) Este objeto encapsula el parser de la librería. Este objeto tan sólo tiene una propiedad, llamada *analizar_input*, que se ocupa de llamar a la función **Parser__parse**, la cual hace todo el trabajo de leer la entrada del jugador, interpretarla y generara las acciones oportunas.

Capítulo 8

Rutinas

8.1. Rutinas de formato del print

Las siguientes rutinas pueden usarse como parte de una línea `print`, como por ejemplo:

```
print "No ves nada especial en ", (el) obj, ".";
```

(al) (*Nuevo*) Imprime el nombre corto del objeto precedido por “*al*” o “*a la*” o “*a los*” o “*a las*”, según cuál sea el artículo requerido para este objeto (ver `(_El)`).

(coge) (*Nuevo*) Imprime la siguiente cadena de caracteres precedida de la palabra “*coge*” o “*toma*”, según el dialecto en uso. Así, por ejemplo `print "No lo ", (coge) "s";` se convierte en “*No lo coges*” o “*No lo tomas*”. Observar que es necesario siempre detrás de este modificador especificar una cadena, aunque sea vacía.

(del) (*Nuevo*) Imprime el nombre corto del objeto precedido por “*del*”, “*de la*”, “*de los*” o “*de las*”, según el artículo requerido para este objeto (ver `(_El)`).

(e) (*Nuevo*) Imprime la letra “*e*” si el objeto es masculino singular, “*a*” si es femenino singular, “*es*” si es masculino plural y “*as*” si es femenino plural. Pensado para construcción de pronombres indirectos como “*les*”, “*las*”. La deducción de género y número se hace siempre en base a los atributos **femenino** y **nombreplural**.

(el) (*Antes (the)*) Imprime el nombre corto del objeto precedido por el artículo definido en minúsculas (véase `(_El)`).

(_El) (*Antes (The)*) Imprime el nombre corto del objeto, precedido por el artículo determinado en mayúsculas. El artículo se tomará de la propiedad *articulos* del objeto, y si no la tuviera se decidirá a partir de su propiedad *genero*, y si tampoco la tuviera se deducirá de sus atributos **femenino** y **nombreplural**. Si tiene el atributo **propio** no llevará artículo (a menos que se haya especificado uno en su propiedad *articulos*).

- (**es**) (*Nuevo*) Escribe “es” o “son”, según el número del objeto.
- (**esta**) (*Nuevo*) Escribe “está” o “están”, según el número del objeto recibido. Esta función es superflua, pues puede usarse en su lugar (n) (véase).
- (**lo**) (*Nuevo*) Escribe “lo”, “la”, “los” o “las” según el valor de los atributos **femenino** y **nombreplural**. Pensado para construir pronombres directos. Esta función es superflua, pues puede usarse en su lugar (o) (véase).
- (**_nombre_**) (Antes (*name*)) Imprime el nombre corto del objeto (si tiene la propiedad *nombre_corto* se usará ésta y si no se usará el nombre suministrado entre comillas en la cabecera del objeto).
- (**MCoge**) (*Nuevo*) Como (coge) pero con la inicial en mayúscula.
- (**MMCoge**) (*Nuevo*) Como (coge) pero con todas las letras en mayúsculas.
- (**n**) (*Nuevo*) Imprime “n” o “” (cadena vacía) según el objeto tenga el atributo **nombreplural** o no lo tenga. Pensado para hacer concordar los verbos con el sujeto, como en `print (_El) ser_vivo, "no parece", (n) ser_vivo, "muy feliz."`.
- (**numero**) (Antes (*number*)) Imprime el siguiente argumento (que será un número o una variable conteniendo un número) como el texto español que representa a ese número. Por ejemplo `print (numero) 123` producirá la salida “*ciento veintitrés*”.
- (**o**) (*Nuevo*) Imprime “o”, “a”, “os” o “as”, según el estado de los atributos **femenino** y **nombreplural** del objeto. Pensado para hacer concordar los adjetivos con el nombre, como en `print (_El) objeto, " ya estaba abiert", (o) objeto, "."`.
- (**s**) (*Nuevo*) Imprime “s” o “” (cadena vacía) según el objeto tenga o no tenga el atributo **nombreplural**. Pensado para hacer concordar algunos adjetivos (que no cambian de género) con el número del objeto. Por ejemplo `print (_El) obj, (es) obj, "verde", (s) obj, "."` que puede producir “*El cajón es verde*”, o “*Los pantalones son verdes*”, o “*La caja es verde*”, según cuál sea el valor de `obj`.
- (**un**) (Antes (*a*)) Imprime el nombre corto del objeto precedido de su artículo indeterminado (“un”, “una”, “unos”, “unas”). Este lo toma de la propiedad *articulos* si el objeto proporciona una, o lo deduce de la propiedad *genero* si el objeto la proporciona. Si no, lo deducirá del estado de los atributos **femenino** y **nombreplural**.

8.2. Rutinas que forman parte del lenguaje

- child** (Antes *child*) Esta rutina recibe como parámetro un objeto y devuelve el primer objeto de los contenidos en él. *Child* en inglés significa ‘hijo’.

children (Antes *children*) Esta rutina recibe como parámetro un objeto y devuelve el número de objetos contenidos en él. *Children* en inglés significa ‘hijos’.

indirect (Antes *indirect*) Esta función recibe varios parámetros: el primero es el nombre de una rutina o un puntero a una rutina; los restantes son lo que se quiera. Lo que hace esta función es llamar a la rutina que recibió como primer parámetro, pasándole como parámetros todos los demás.

metaclass (Antes *metaclass*) Esta rutina recibe como parámetro un identificador y devuelve de qué tipo es. Puede devolver uno de los siguientes valores: **string**, **class**, **object** o **routine**.

parent (Antes *parent*) Esta rutina recibe como parámetro un objeto y devuelve el objeto dentro del cual (o sobre el cual) está contenido. *Parent* en inglés significa ‘padre’.

Print__PName (Antes *Print__PName*) Esta rutina no está programada en la librería, aunque podría proporcionarse otra que la sustituyera, pero debería conservar su nombre original en inglés, ya que es la rutina que se invocará al especificar el formato `print (property) x` para imprimir nombres de propiedades.

random (Antes *random*) *Random* en inglés significa ‘aleatorio’. Esta función tiene dos formas:

- `random (n)` Retorna un número entero comprendido entre 1 y n.
- `random (3, 6, 8, 10)` Retorna uno cualquiera de esos números (no han de ser esos precisamente). En lugar de números pueden usarse también cadenas de caracteres, como por ejemplo `random ("rojo", "verde", "azul")`.

sibling (Antes *sibling*) Esta rutina recibe como parámetro un objeto y devuelve su siguiente *hermano* (todos los objetos contenidos dentro de otro son *hermanos*). *Sibling* en inglés significa hermano.

8.3. Rutinas proporcionadas por la librería

ActualizarPronombre (Antes *PronounNotice*) Esta función recibe como parámetro un objeto y se ocupa de actualizar los pronombres para que se refieran a este objeto (de modo que en lo sucesivo “-lo” se referirá a él). En realidad el pronombre actualizado será “-lo”, “-la”, “-los” o “-las” dependiendo del valor de los ¿? atributos *genero* y **nombrequplural** de ese objeto, los cuales a su vez pueden depender de qué palabra haya usado el jugador para referirse a este objeto (es decir, si la palabra ha sido hallada en la propiedad *nombre_m*, *nombre_f*, *nombre_mp* o *nombre_fp*).

ArrancarDaemon (Antes *StartDaemon*) Esta función recibe como parámetro un objeto y pone en marcha el *daemon* de dicho objeto. Como consecuencia, la propiedad *daemon* de este objeto será llamada al final de cada turno.

ArrancarReloj (Antes *StartTimer*) Esta función recibe dos parámetros. El primero es un objeto y el segundo es un tiempo. La librería almacenará en la propiedad *tiempo_restante* el tiempo que le hemos suministrado. Al final de cada turno restará 1 al tiempo restante de este objeto y si se alcanza el valor cero, llamará a la rutina *tiempo_agotado* de este objeto.

BucleAlcance (Antes *LoopOverScope*) Esta función recibe dos parámetros. El primero es el nombre de otra rutina (llamémosla *R*, y el segundo (que es opcional) es un objeto (normalmente un personaje, llamémosle *actor*). Lo que hace esta función es llamar repetidas veces a la rutina *R* pasándole como parámetro cada uno de los objetos que estarían al alcance del *actor* en cuestión. Si el segundo parámetro se omite, tomará el valor por defecto del jugador.

BuscarEnDiccionario (Antes *DictionaryLookup*) Esta función recibe dos parámetros. El primero es un vector de caracteres (que se supone que contiene una palabra) y el segundo es la longitud de dicho vector. La función devuelve la dirección de diccionario de la primera palabra del vector, y si no la encuentra en el diccionario devolverá cero.

CambiarDefecto (Antes *ChangeDefault*) Recibe dos parámetros: el nombre de una propiedad y un valor. Lo que hace es cambiar el valor por defecto de la propiedad dada (excepto si la propiedad es *nombre* o *nombre_m*).

CambiarJugador (Antes *ChangePlayer*) Recibe dos parámetros: un objeto y un entero que puede ser 1 ó 0. El objeto debe tener la propiedad *cantidad* y a partir de ese momento representará al jugador. El segundo parámetro indica si queremos que aparezca entre paréntesis en el título de las habitaciones la nueva personalidad del jugador. Si ponemos 0 (o no damos este segundo parámetro) esta aclaración no saldrá. La rutina en sí misma no imprime nada.

CDefArt (Antes *CDefArt*) Rutina que imprime el nombre corto de un objeto precedido de su artículo definido en mayúsculas. *CDefArt* (obj) equivale a `print (_El) obj`. Esta rutina debe conservar su nombre original en inglés porque es la invocada cuando se usa la cadena de formato `print (The) obj`.

CompararSinSigno (Antes *UnsignedCompare*) Recibe dos parámetros *x* e *y* y devuelve 0 si $x = y$, 1 si $x > y$ y -1 si $x < y$. Normalmente no es necesario llamar a esta función para hacer estas comparaciones y basta con usar los operadores de comparación, a menos que las cantidades a comparar representen direcciones dentro de la máquina Z y deban por tanto ser interpretadas como enteros sin signo.

Conseguido (Antes *Achieved*) Recibe un parámetro que es el número de tarea conseguida. El juego llamará a esta función cada vez que el jugador haya superado con éxito uno de los objetivos del juego. La librería aumentará su puntuación y anotará que dicha tarea ya está terminada (para informar de ello ante el comando PUNTUACION TOTAL).

DefArt (Antes *DefArt*) Rutina que imprime el nombre de un objeto precedido de su artículo definido. `DefArt (obj)` equivale a `print (el) obj`. Esta rutina debe conservar su nombre original en inglés porque es la invocada cuando se usa la cadena de formato `print (the) obj`.

DentroDelAlcance (Antes *ScopeWithin*) Recibe como parámetro un objeto. La librería añadirá este objeto a su lista de objetos al alcance (y también todos los objetos contenidos en él que de acuerdo con las reglas estándar deberían estar también al alcance).

DibujarLineaEstado (Antes *DrawStatusLine*) Pinta la barra de estado en lo alto de la pantalla. Por defecto imprime a la izquierda el título de la habitación y a la derecha la puntuación y número de turnos usados. La idea es que el programador puede definir su propia rutina `DibujarLineaEstado` que sustituya a esta (debe anunciar la intención de hacer esto escribiendo `Replace DibujarLineaEstado;` antes de incluir `EParser.h`).

DireccionDePalabra (Antes *WordAddress*) Recibe un número que representa el número que hace una de las palabras que el jugador ha escrito. Por ejemplo, si el jugador pone “saca la espada de la funda”, el número 3 representaría la palabra “espada”, pues es la tercera palabra. La función devuelve una dirección de memoria, que es la dirección donde está almacenada la primera letra de dicha palabra (esta dirección estará dentro del buffer de parsing, el cual es la variable *buffer*).

DoMenu (Antes *DoMenu*) Esta rutina está relacionada con la creación de menús.

DominioNombre (Antes *NounDomain*) Esta función recibe dos objetos como parámetros. El primero suele ser el jugador y el segundo el lugar en que se halla. Lo que hace es buscar en esos objetos (siguiendo las reglas de alcance estándar) si hay alguno cuyo nombre pudiera encajar con lo que ha escrito el jugador (para saber lo que ha escrito el jugador usará la función `SiguientePalabra`, la cual a su vez usa la variable *np*). Si además le pasamos un tercer parámetro igual a 1, limitará su búsqueda a objetos poseídos por el jugador, y si es igual a 2, la limitará a criaturas animadas.

Devuelve 0 si no encuentra nada que encaje, la constante **REPARSE_CODE** si hay que reinterpretar toda la entrada desde cero (normalmente porque ha hecho una pregunta de aclaración al jugador y ha obtenido un nuevo comando) o el valor del objeto que mejor encaja.

ElementoCualquiera (Antes *RandomEntry*) Recibe como parámetro una tabla y devuelve un elemento cualquiera de la misma. Es útil para generar un mensaje aleatorio a elegir entre varios.

EnglishNumber (Antes *EnglishNumber*) Imprime un número como texto (en español, a pesar del nombre de la función). `EnglishNumber (cantidad)` equivale a `print (numero) cantidad`. Esta rutina debe conservar su nombre original en inglés porque es la invocada cuando se usa la cadena de formato `print (number) x`.

EscribirListaDesde (Antes *WriteListFrom*) Esta rutina es básica para crear inventarios o listas de objetos. Recibe dos parámetros, el primer parámetro es un objeto que es a su vez el primero de los contenidos en otro. La rutina imprimirá el nombre de este objeto y el de todos sus hermanos y si cualquiera de ellos tiene otros objetos dentro, puede imprimirlos también. El segundo parámetro es un número que indica el estilo deseado para la lista. Este número se compone sumando las constantes descritas en la sección 9.4 en la página 99.

FijarPronombre (Antes *SetPronoun*) Permite asignar un valor directamente a uno de los pronombres. Recibe como primer parámetro el pronombre, y como segundo parámetro el objeto al que queremos que se refiera. Por ejemplo *FijarPronombre* ('-lo', baston);

HayLuz (Antes *OffersLight*) Recibe como parámetro un objeto y retorna **true** si en el objeto hay luz (porque contiene algún objeto luminoso). Maneja correctamente recipientes transparentes.

ImprimirOExecutar (Antes *PrintOrRun*) Recibe dos parámetros. El primero es un objeto y el segundo el nombre de una propiedad. Lo que hace es comprobar si el objeto en cuestión presenta dicha propiedad. Si la posee y es una cadena de caracteres o un número, imprimirá su valor. Si la posee y es una rutina, ejecutará la rutina. Si no la posee no hará nada. Es la forma más segura y genérica de imprimir una propiedad de un objeto y es muy usada en diferentes partes de la librería de acciones.

IndefArt (Antes *InDefArt*) Rutina que imprime el nombre de un objeto precedido de su artículo indefinido. *InDefArt* (obj) equivale a `print (un) obj`. Esta rutina debe conservar su nombre original en inglés porque es usada cuando se especifica la cadena de formato `print (a) obj`.

IntentarNumero (Antes *TryNumber*) Recibe como parámetro un entero que representa el número que hace una de las palabras que el jugador ha escrito. Por ejemplo, si el jugador pone “saca la espada de la funda”, el número 3 representaría la palabra “espada”, pues es la tercera palabra. La función intenta interpretar esa palabra como si fuera un número en español (sólo comprende desde “uno” hasta “veinte”). Si lo logra, devolverá el valor de ese número, si no devolverá -1000. También entiende los números escritos con cifras (como “123”), pero no pueden ser superiores a 10000.

JugadorA (Antes *PlayerTo*) Esta rutina sirve para mover el jugador a otro lugar y debe usarse siempre en lugar de moverlo directamente con `move jugador to lugar`, ya que usando esta función de la librería nos garantizamos que se actualizan correctamente las puntuaciones, los atributos de visita, etc. Recibe como parámetro el lugar al que queremos mover el jugador. Podemos especificar en el segundo parámetro opcional el valor 1 si queremos impedir que se describa el lugar al que acaba de llegar.

Local (Antes *Locale*) Esta función se ocupa de listar los contenidos de un objeto como si ese objeto fuera una habitación (de hecho esta función es llamada ante la acción *Mirar*, para que muestre los contenidos de la habitación). Primero se describen

los objetos que tienen su párrafo propio (es decir, aquellos que proporcionan una propiedad *inicial*, *describir*, *si_abierto*, *si_cerrado*, *si_encendido* o *si_apagado*) y seguidamente en una lista todos los demás objetos. La función recibe tres parámetros: el primero es la localización cuyos contenidos queremos listar, el segundo es un texto de cabecera a mostrar delante de la lista final en el caso de que no haya objetos con “párrafo propio” (por ejemplo el texto podría ser "Puedes ver: " y el tercero el texto a mostrar en caso de que sí haya objetos con párrafo propio (por ejemplo `.Además, puedes ver:` ")

LongitudDePalabra (Antes *WordLength*) Recibe como parámetro un entero que representa el número que hace una de las palabras que el jugador ha escrito. Por ejemplo, si el jugador pone “saca la espada de la funda”, el número 3 representaría la palabra “espada”, pues es la tercera palabra. Retorna el número de letras de esa palabra (6 en nuestro ejemplo)

ObjetoEsIntocable (Antes *ObjectIsUntouchable*) Esta función es usada constantemente por la librería pues para la mayoría de las acciones se comprueba antes si el jugador puede tocar el objeto al que se refiere. Recibe como parámetro el objeto a comprobar y retorna **true** si existe algún tipo de barrera sólida entre el jugador y el objeto (por ejemplo un recipiente cerrado). Además escribe un mensaje indicando el problema (por ejemplo: “No puedes porque la botella está cerrada”). Si queremos evitar el mensaje podemos especificar en un segundo parámetro opcional el valor 1.

PalabraEnPropiedad (Antes *WordInProperty*) Esta función recibe tres parámetros: el primero es una palabra (de diccionario), el segundo es un objeto y el tercero es una propiedad (que debe ser una lista de palabras de diccionario, como por ejemplo la propiedad *nombre*). Retorna **true** si la palabra dada está contenida en esa lista para ese objeto.

PararDaemon (Antes *StopDaemon*) Recibe como parámetro un objeto y detiene su *daemon* (es decir, la rutina *daemon* de ese objeto ya no será llamada al final de cada turno, a menos que se active de nuevo).

PararReloj (Antes *StopTimer*) Recibe como parámetro un objeto, y detiene su cuenta atrás (es decir, su propiedad *tiempo_restante* ya no será decrementada al final de cada turno)

PermitirEmpujarDir (Antes *AllowPushDir*) No recibe parámetros. Es llamada normalmente desde la rutina *antes* de un objeto que ha capturado la acción *EmpujarDir*, para indicar a la librería que admite el empujón. Esta función moverá al objeto en la dirección requerida (siempre que no haya nada que lo impida).

PonerAlAlcance (Antes *PlaceInScope*) Recibe un objeto como parámetro y lo añade a la lista de objetos al alcance. Suele usarse desde dentro de la propiedad *suma_al_alcance* de los objetos para añadir otros al alcance.

PonerLaHora (Antes *SetTime*) Recibe dos parámetros, el primero es un contador de tiempo (en minutos) y el segundo es la velocidad con la que pasa el tiempo (llamémosle *n*) . Lo que hace es cambiar el valor de la variable *la_hora* dándole como

valor el primer parámetro. Además actualizará esta variable al final de cada turno en la forma siguiente: si n era 0, la hora no cambiará; si era positiva, el tiempo aumentará en n segundos al final de cada turno, si era negativa la librería esperará $-n$ turnos antes de incrementar el tiempo en un minuto.

PrintShortName (Antes *PrintShortName*) Imprime el nombre corto de un objeto. `PrintShortName (obj)` es equivalente a `print (_nombre_) obj`. El nombre de esta función se ha traducido (debe ir obligatoriamente en inglés porque es la función que el lenguaje usa cuando encuentra la especificación de formato `print (name) obj`).

PruebaDeAlcance (Antes *TestScope*) Recibe dos parámetros, el primero es un objeto y el segundo un personaje. Devuelve **true** si el objeto está al alcance del personaje (si se omite el segundo parámetro el personaje será el propio jugador).

RutinasDespues (Antes *AfterRoutines*) Ejecuta la rutina despues del objeto que ha intervenido en la acción (en algunas acciones también llama a la rutina despues del *otro* objeto).

SiguientePalabra (Antes *NextWord*) No recibe parámetros. Cada vez que se llama a esta función retorna una palabra (de diccionario) de las que ha escrito el jugador y avanza a la siguiente. Utiliza la variable **np** para saber cuál es el número de palabra que toca. Si cambiamos desde fuera el valor de **np** cambiaremos también cuál será la siguiente palabra que retornará esta función. Recordemos que si una de las palabras escritas por el jugador no estaba en el diccionario su valor de diccionario será cero (y este será el valor retornado por esta función en ese caso). También devuelve 0 si no quedan más palabras por examinar.

SiguientePalabraParar (Antes *NextWordStopped*) Idéntico a **SiguientePalabra** con la diferencia de que devuelve -1 en el caso de que se hayan agotado las palabras.

SiNo (Antes *YesOrNo*) Tras plantear una pregunta del tipo “¿Estás seguro?”, se llama a esta función, la cual espera hasta que el jugador haya respondido «Sí» o «No» (insistiéndole si responde otra cosa).

SumarAlAlcance (Antes *AddToScope*) Recibe como parámetro un objeto y lo añade a la lista de objetos al alcance (si el objeto que ha llamado a esta función estaba a su vez en el alcance).

Teclado (Antes *Keyboard*) Recibe dos parámetros: un vector de caracteres y una tabla de parsing. Esta función espera a que el jugador escriba algo, y guarda el texto recibido en el vector de caracteres que recibió como primer parámetro. Además, hace una mini-interpretación consistente en buscar las palabras en el diccionario y guarda el resultado en la tabla de parsing usada como segundo parámetro.

Esta función es usada por la librería, pero el programador puede llamarla también cuando quiera obtener una entrada desde el teclado que no pase por el parsing.

TieneFuenteDeLuz (Antes *HasLightSource*) Recibe como parámetro un objeto y retorna **true** si ese objeto emite luz (ya sea por sí mismo o porque la emite otro objeto en el interior de éste y visible a través de él).

ValorDelPronombre (Antes *PronounValue*) Recibe como parámetro un pronombre y devuelve el objeto asociado con él (por ejemplo `ValorDelPronombre ('-lo')`). El valor devuelto será **NULL** si ese pronombre aún no representaba nada.

ValorOEjecutar (Antes *ValueOrRun*) Recibe como parámetros en primer lugar un objeto y en segundo una propiedad. La función comprueba si el objeto tiene esa propiedad y si la tiene y es una rutina la ejecuta, y si no es una rutina retorna su valor. En el caso de que fuera una rutina, devolverá el valor devuelto a su vez por ésta.

ZRegion (Antes *ZRegion*) Recibe una variable cualquiera e intenta adivinar a qué tipo de objeto apunta. Retornará 1 si se trata de un objeto, 2 si es una dirección de una rutina o 3 si es una dirección de una cadena. Si retorna 0 es que no ha podido adivinarlo.

8.4. Rutinas que deben programarse en el juego

Estas funciones también son conocidas con el nombre de *puntos de entrada*.

AlAlcance (Antes *InScope*) Si existe es llamada durante el parsing, recibiendo como parámetro el **actor**. Esta rutina podrá añadir a la lista de cosas al alcance cualquier otro objeto que desee añadir (puede usar para ello la función `SumarAlAlcance`). Si además, retorna **true**, la librería no añadirá nada más al alcance, si en cambio retorna **false**, la rutina añadirá los objetos que según su propio criterio estén al alcance del actor.

AntesDelParsing (Antes *BeforeParsing*) Si existe, esta función es llamada tras leer lo que el jugador ha escrito, y tras convertirlo a “informés”, (esta transformación consiste en separar los pronombres de los verbos, por ejemplo “MIRALO” se convierte en “MIRA -LO”, y en eliminar algunas contracciones, por ejemplo “AL” se convierte en “A”, y “DEL” se convierte en “DE”, además de quitar los acentos de todas las palabras que no hayan sido halladas en el diccionario). Cuando esta función es llamada, aún no se ha empezado el parsing, por lo que podría cambiarse lo tecleado por el jugador en otra cosa (no obstante, hacer esto exige una programación de bastante bajo nivel sobre el vector **buffer** que contiene el texto tecleado).

CaminarAOscuras (Antes *DarkToDark*) Si existe esta función será llamada cuando el jugador se mueve desde un lugar que no tenía luz a otro lugar que tampoco tiene luz.

Curiosidades (Antes *Amusing*) Si está definida esta función y también la constante **HAY_CURIOSIDADES**, será llamada una vez que el jugador haya ganado el juego.

ElegirObjetos (Antes *ChooseObjects*) Esta rutina recibe dos parámetros. El primero es un objeto y el segundo un número que puede valer 0, 1 ó 2, según el motivo por el que se llama a esta función. Los motivos pueden ser:

- 0 El parser está preguntando si este objeto debe ser excluido de la lista de “todos” los objetos.
- 1 El parser está preguntando si este objeto debe ser incluido en la lista de “todos” los objetos.
En los dos casos anteriores la rutina devolverá 0 para dejar ese objeto en la lista, 1 para obligar a que se incluya ó 2 para obligar a que se excluya.
- 2 El parser tiene dudas entre varios objetos, y está preguntando si este objeto podría ser el que el jugador solicita. En este caso la rutina deberá mirar lo que ha escrito el jugador y decidir si es aplicable a ese objeto, retornando una puntuación entre 0 y 9 según sea nada aplicable o muy aplicable.

ErrorParser (Antes *ParserError*) Si existe, se llamará para que imprima mensajes de error alternativos. Recibe como parámetro el número del error. Si no existe (o devuelve **false**), la librería mostrará uno de los errores estándares. Otra forma más simple de cambiar los errores estándar consiste en definir el objeto **MensajesLibreria**.

ImprimirRango (Antes *PrintRank*) Si existe, es llamada después de imprimir la puntuación del jugador (para que añada algo como “*lo que te otorga la categoría de Maestro*”).

ImprimirTareas (Antes *PrintTaskName*) Si existe será llamada pasándole como parámetro un número de tarea, para que imprima su nombre. Véase **HAY_TAREAS**, **NUMERO_TAREAS**, *puntuacion_tareas* y Conseguido.

ImprimirVerbo (Antes *PrintVerb*) Es llamada cuando hay que imprimir la forma infinitiva de un verbo. Normalmente esto ya lo hace la rutina *IdiomaVerbo*, pero si el verbo fuese extraño y no saliera correctamente, se le da una oportunidad a esta función para que lo corrija.

Inicializar (Antes *Initialise*) Esta debe existir en todo juego, ya que es lo primero que la librería ejecutará. Lo habitual aquí es inicializar la variable **localizacion** con el lugar donde empieza el jugador, y escribir algún mensaje introductorio.

InterpretarNombre (Antes *ParseNoun*) Rutina encargada de interpretar el nombre de un objeto. Recibe un objeto y debe decidir cuántas palabras seguidas de las escritas por el jugador son aplicables a este objeto. Esta rutina, aunque era un *punto de entrada* en la librería original, en **InformATE!** es utilizada para hacer el *parsing* usando las nuevas propiedades *adjetivos*, *nombre_m*, *nombre_f*, *nombre_mp* y *nombre_fp*, por lo que no debe (re)definirse esta función en los juegos.

InterpretarNumero (Antes *ParseNumber*) Si existe, el parser la llamará para que interprete lo escrito por el jugador cuando espera que se trate de un número, de acuerdo con las especificaciones de la gramática.

LugarNuevo (Antes *NewRoom*) Si existe es llamada cada vez que la *localizacion* cambia, antes de imprimir ninguna otra cosa.

MasAlla (Antes *AfterLife*) Si está definida, será llamada cuando el *jugador* haya muerto. Desde esta función se puede cambiar la variable *banderafin* y si se le da el valor cero, el jugador habrá resucitado.

MensajeMuerte (Antes *DeathMessage*) Si existe y *banderafin* es mayor o igual a 3, la librería llamará a esta rutina para que imprima un mensaje (explicando la causa de la muerte del jugador). Esta función puede consultar *banderafin* para elegir entre varios mensajes posibles.

MirarHaciaSub (*Nuevo*) Si existe, se usará esta función para imprimir el mensaje por defecto cuando el jugador mire hacia el norte, sur, etc. Si no existe, el mensaje por defecto será “No ves nada más digno de mención al mirar hacia el norte”. La rutina recibirá como parámetro el objeto-dirección hacia el que el jugador quiere mirar (*obj_n*, *obj_s*, etc.)

PasaElTiempo (Antes *TimePasses*) Si existe es llamada al final de cada turno.

PreguntarPreposicion (*Nuevo*) Si existe es llamada cuando la librería debe imprimir una preposición como inicio de pregunta. Por ejemplo “¿Por dónde quieres mirar?”. La preposición depende del verbo y de la línea de gramática que se haya interpretado, ya que por ejemplo a una línea que espere “mirar a” le iría bien la pregunta “¿Hacia dónde quieres mirar?”, mientras que otra que espere “Preguntar a”, le iría mejor “¿A quién quieres preguntar?”, a pesar de que ambas usan la preposición “a”.

Normalmente esta rutina no es necesaria ya que la librería maneja correctamente la mayor parte de las preposiciones, pero si se define algún verbo nuevo para el que la respuesta estándar de la librería no encajara, esta rutina da una oportunidad de mejorarla.

Para que esta rutina pueda decidir cuál es la forma más adecuada de iniciar la pregunta, recibe como parámetros el verbo y la preposición esperados por la gramática.

RutinaMirar (Antes *LookRoutine*) Si existe será llamada al final de toda acción *Mirar* (tanto las originadas por la petición MIRA del usuario, como las originadas al entrar en una habitación).

RutinaPostJuego (Antes *GamePostRoutine*) Si existe, es llamada después de haber efectuado todas las acciones, en cada turno.

RutinaPreJuego (Antes *GamePreRoutine*) Si existe, es llamada en cada turno, tras haber interpretado el comando del jugador pero antes de efectuar ninguna acción (antes incluso de llamar a los procedimientos *antes* de los objetos).

TrasElPrompt (Antes *AfterPrompt*) Si está definida será llamada justo después de haber impreso el *prompt* (el cual normalmente es “>”, pero puede ser cambiado desde el objeto **MensajesLibreria**).

VerboDesconocido (Antes *UnknownVerb*) Si existe es llamada cuando el parser esperaba un verbo pero encuentra una palabra desconocida.

8.5. Rutinas usadas internamente por la librería

Estas funciones son usadas por la librería y no están pensadas para ser usadas por el programador, aunque puede hacerlo. El único problema es que no están completamente documentadas. El programador interesado puede examinar el código fuente para ver qué hacen.

ActualizarEstatus (Antes *DisplayStatus*)

AjustarLuz (Antes *AdjustLight*)

AnalizarToken (Antes *AnalyseToken*)

AniadirMulti (Antes *MultiAdd*)

AnotarLlegada (Antes *NoteArrival*)

AnotarObjetosObtenidos (Antes *NoteObjectAcquisitions*)

AntepasadoComun (Antes *CommonAncestor*)

Anuncio (Antes *Banner*)

AveriguarPrimeraPreposicion (*Nuevo*)

AveriguarPreposicion (*Nuevo*)

BorrarDescriptores (Antes *ResetDescriptors*)

BuscarEnAlcance (Antes *SearchScope*)

BuscarEntreVerbosIrregulares (*Nuevo*)

CadenaDePreposiciones (Antes *PrepositionChain*)

CientosEspanol (*Nuevo*)

ContieneIndirectamente (Antes *IndirectlyContains*)

CopiarBuffer (Antes *CopyBuffer*)

DeboBajarRecurso (Antes *WillRecurse*)

DecirQueHaySobre (Antes *SayWhatsOn*)

DentroDelAlcance_O (Antes *ScopeWithin_O*)

DepurarAccion (Antes *DebugAction*)

DepurarAtributo (Antes *DebugAttribute*)

DepurarLineaGramatica (Antes *DebugGrammarLine*)

DepurarParametro (Antes *DebugParameter*)

DepurarToken (Antes *DebugToken*)

Descriptores (Antes *Descriptors*)

DesempaquetarLineaGramatica (Antes *UnpackGrammarLine*)

DigitoEspanol (*Nuevo*)

DirDicc__Num (Antes *Dword__No*)

EfectuarAccionesAlcance (Antes *DoScopeAction*)

EjecutarRutinas (Antes *RunRoutines*)

EjecutarVida (Antes *RunLife*)

EligeObjetos (*Nuevo*)

EliminarDuplicados (*Nuevo*)

EnglishNumber (Antes *EnglishNumber*)

ErrorDeEjecucion (Antes *RunTimeError*)

EscribeAntesElemento (Antes *WriteBeforeEntry*)

EscribeListaR (Antes *WriteListR*)

EscribeTrasElemento (Antes *WriteAfterEntry*)

EscribirListaDesde (Antes *WriteListFrom*)

EspanolAInformes (*Nuevo*)

FiltrarMulti (Antes *MultiFilter*)

FiltroUsuario (Antes *UserFilter*)

HacerEncaje (Antes *MakeMatch*)

Identicos (Antes *Identical*) Esta rutina decide si dos objetos pueden ser distinguidos por el jugador a la hora de nombrarlos. Lo que hace es comparar, una por una y en este orden, las listas de palabras que aparecen en las propiedades *nombre*, *adjetivos*, *nombre_f*, *nombre_mp* y *nombre_fp* de ambos objetos. Si todas son iguales en los dos objetos, considera que estos son indistinguibles por el jugador.

IdiomaAInformes (Antes *LanguageToInformese*)

IdiomaContraccion (Antes *LanguageContraction*)

IdiomaDirecciones (Antes *LanguageDirection*)

IdiomaEsVerbo (Antes *LanguageIsVerb*)

IdiomaHoraDelDia (Antes *LanguageTimeOfDay*)

IdiomaImprimirComando (Antes *LanguagePrintCommand*)

IdiomaImprimirNombreCorto (Antes *LanguagePrintShortName*)

IdiomaNumero (Antes *LanguageNumber*)

IdiomaPreguntarPrep (*Nuevo*)

IdiomaVerbo (Antes *LanguageVerb*)

ImpNumAlin (Antes *PANum*)

Imprimir__Espacios (Antes *Print__Spaces*)

ImprimirComando (Antes *PrintCommand*)

ImprimirIrregular (*Nuevo*)

IniciarPregunta (*Nuevo*)

InsertarLetraBuffer (*Nuevo*)

IntentarCogerObjeto (Antes *AttemptToTakeObject*)

IntentarEncajarObjeto (Antes *TryGivenObject*)

InterpretarToken (Antes *ParseToken*)

ListarIguales (Antes *ListEqual*)

LowKey_Menu (Antes *LowKey_Menu*)

Lugares1Sub (Antes *Places1Sub*)

M__L (Antes *L__M*)

M__L (Antes *L__M*)

Main (Antes *Main*)

MejorIntuicion (Antes *BestGuess*)

MLIdioma (Antes *LanguageLM*)

MoverJugador (Antes *MovePlayer*)

MoverObjetosFlotantes (Antes *MoveFloatingObjects*)

MoverPalabra (Antes *MoveWord*)

NoLoVeo (Antes *CantSee*)

NoSoportaAlJugador (Antes *NotSupportingThePlayer*)

NotificarLaPuntuacion (Antes *NotifyTheScore*)

Num__DirDicc (Antes *No__Dword*)

ObjetoEnAlcancePorAlgo (Antes *ObjectScopedBySomething*)

ObjetoEsIntocable (Antes *ObjectIsUntouchable*)

Objetos1Sub (Antes *Objects1Sub*)

ObtenerGenero (Antes *GetGender*)

ObtenerGNADeObjeto (Antes *GetGNAOfObject*)

OrdenarJuntos (Antes *SortTogether*)

OrdenarLista (Antes *SortOutList*)

PalabraNumero (Antes *NumberWord*)

PalabraSustantivo (Antes *NounWord*)

Parser__parse (Antes *Parser__parse*)

PonerArticuloDelante (Antes *PrefaceByArticle*)

Print_ScL (Antes *Print_ScI*)

PSN__ (Antes *PSN__*)

PuntuacionListaEncajes (Antes *ScoreMatchL*)

PuntuacionLlegada (Antes *ScoreArrival*)

QuitandoRFinal (*Nuevo*)

QuitarAcentos (*Nuevo*)

R_Process (Antes *R_Process*) El nombre de esta función debe mantenerse en el inglés original. Esta rutina es la que se ocupa de ejecutar una acción cuando es invocada desde dentro del programa con una sintaxis como <<Accion>>.

ReiniciarPalabrasVagas (Antes *ResetVagueWords*)

RevisarMulti (Antes *ReviseMulti*)

RutinasAntes (Antes *BeforeRoutines*)

RutinasDespues (Antes *AfterRoutines*)

SeRefiere (Antes *Refers*)

SeVeATraves (Antes *IsSeeThrough*)

SiguienteElemento (Antes *NextEntry*)

SustraerMulti (Antes *MultiSub*)

TestCriatura (Antes *CreatureTest*)

Tokenise__ (Antes *Tokenise__*)

TopeAlcanzable (Antes *ScopeCeiling*)

TrazarAccion (Antes *TraceAction*)

UnaDireccion (Antes *ADirection*)

XCompruebaMover (Antes *XTestMove*)

XObj (Antes *XObj*)

Capítulo 9

Constantes

9.1. Constantes definibles por el juego

ADMITIR_COMANDO_SALIDAS (*Nuevo*) Si esta constante está definida antes de incluir los ficheros de la librería, entonces se definirá la acción “salidas” (abreviatura “x”) que muestra al jugador la lista de salidas obvias del lugar en que se halle. Si la constante no está definida, el verbo “salidas” será desconocido para el juego. No es necesario asignarle ningún valor, basta definirla.

ADMITIR_INFINITIVOS (*Nuevo*) Esta constante ya no está presente en la librería, pues los infinitivos son admitidos por defecto en la actual versión de InformATE!.

El mecanismo para “comprender” un verbo en infinitivo consiste en primero quitarle la ‘r’ final y comparar el resultado con los verbos conocidos (de este modo, ‘TIRAR’ se convierte en ‘TIRA’ que es conocido, ‘EMPUJAR’ en empuja, etc.) Pero este método no vale para los irregulares, ya que ‘ABRIR’ se convertiría en ‘ABRI’ que no es conocido, por lo que la librería intentará un segundo método que consiste en comparar el verbo escrito por el jugador con toda la lista de verbos irregulares definida. Si se encuentra, se cambia el verbo escrito por el jugador por la primera palabra de la propiedad *imperativo* de ese verbo irregular. Así, ‘ABRIR’ se convertiría en ‘ABRE’, que ya es conocido.

Cabe tener en cuenta que antes de intentar todo esto, la librería ya ha detectado la existencia de un sufijo extra en el verbo (como en ‘EXAMINARLO’) y lo ha separado del mismo (creando el comando ‘EXAMINAR -LO’, por lo que el método de quitar la ‘r’ final es válido incluso en estos casos.

DEBUG (Antes *DEBUG*) Si esta constante está definida antes de incluir la librería (no es necesario darle valor, basta definirla), entonces el juego comprenderá todos los verbos de depuración. Son verbos que permiten “hacer trampa” en el juego, obteniendo cualquier objeto, permitiendo el viaje a cualquier lugar, etc.

HAY_CURIOSIDADES (Antes *AMUSING_PROVIDED*) El programador debe definir ésta si va a proporcionar la función llamada *Curiosidades*. Si esta constante está

definida, cuando el jugador *gane* el juego se le preguntará si desea conocer algunas curiosidades sobre el mismo. Si así lo solicita, la función *Curiosidades* será llamada (normalmente imprimirá cómo acceder a algunos secretos o partes graciosas del juego).

HAY_TAREAS (Antes *TASKS_PROVIDED*) El programador debe definir ésta si desea definir una serie de “objetivos clave” que el jugador debe ir cumpliendo en el juego, de forma que cuando los logre, su puntuación aumentará, y cuando el jugador ponga «PUNTUACIÓN DETALLADA» se le explique por qué ha conseguido cada punto (con una frase como por ejemplo “5 puntos por deshacerte del perro”).

Naturalmente, además de definir esta constante el programador debe proporcionar más detalles. En particular debe almacenar en la constante **NUMERO_TAREAS** el número de “objetivos” a cumplir. Debe crear un *array* llamado *puntuacion_tareas* cuyos elementos son los puntos conseguidos por completar cada tarea. Y debe escribir una rutina llamada *ImprimirTareas* que escriba el texto explicativo para cada tarea (en el ejemplo anterior debería escribir “por deshacerte del perro”). Esta función recibe un parámetro, que es el número de tarea cuyo texto explicativo debe mostrar.

Historia (Antes *Story*) Nombre del juego, su título. Por ejemplo “El Despertar”.

INFIX (Antes *INFIX*) Si esta constante está definida antes de incluir la librería (no es necesario darle valor, basta definirla), entonces el juego comprenderá todos los verbos de depuración, más los especiales del modo *infix*.

LLEVAR_MAX (Antes *MAX_CARRIED*) Máximo número de objetos que el jugador puede transportar en sus manos. Si uno de los objetos que lleva es un recipiente con suficiente capacidad, podrá meter muchas cosas en él y en este caso cuenta como un solo objeto. Si no se define esta constante tomará el valor por defecto 100.

MAX_RELOJES (Antes *MAX_TIMERS*) Máximo número de relojes temporizadores que pueden estar activos simultáneamente (ver la propiedad *tiempo_agotado*). Su valor por defecto es 32 y no debe cambiarse, a menos que durante la fase de pruebas del juego aparezca un mensaje de error diciendo que se haga.

NO_LUGARES (Antes *NO_PLACES*) Si se define esta constante antes de incluir la librería, no se añadirán al juego las acciones *Lugares* y *Objetos*, que muestran los lugares que ha visitado el jugador y los objetos que ha llevado alguna vez, respectivamente.

NO_PUNTUACION (*Nuevo*) Si se define antes de incluir la librería, el juego no mostrará de ningún modo la puntuación y no estarán presentes las rutinas que la muestran (como *PuntuacionSub*), con lo que se reducirá el tamaño del binario del juego. Las variables y constantes que guardan la puntuación podrán seguir usándose, por lo que esta constante se puede usar también para eliminar temporalmente la puntuación de un juego que hacía uso de ella.

NUMERO_TAREAS (Antes *NUMBER_TASKS*) Número de “objetivos clave” que el jugador debe lograr a lo largo del juego. Véase la constante **HAY_TAREAS**.

OBJETO_SACO (Antes *SACK_OBJECT*) El programador puede definir esta constante asignándole como valor un objeto del juego (que debe ser un recipiente). Si lo hace así, entonces la librería usará ese objeto como una “mochila”, es decir, si el jugador lleva este objeto consigo y al ir recogiendo cosas sus manos se llenan (alcanza el tope marcado por **LLEVAR_MAX**), la librería automáticamente moverá esos objetos al interior del objeto saco (informando al jugador de ello con un mensaje como “[*primero meto la piedra en la mochila para hacer sitio*]”).

PUNTOS_LUGAR (Antes *ROOM_SCORE*) Si el jugador entra en una habitación que tenga el atributo **valepuntos**, su puntuación aumentará en tantos puntos como indica esta constante. Por defecto vale 5.

PUNTOS_OBJETO (Antes *OBJECT_SCORE*) Si el jugador coge un objeto que tenga el atributo **valepuntos**, su puntuación aumentará en tantos puntos como indica este constante. Por defecto vale 4.

PUNTUACION_MAX (Antes *MAX_SCORE*) Máximo de puntuación que puede lograrse en el juego. El programador debe calcular *a mano* esta puntuación y asignarla a esta variable. Es lo que sale cuando al abandonar se muestra el mensaje “*Has logrado 0 puntos de un total de 120*”. Por defecto vale 0.

Release (Antes *Release*) El nombre de esta constante permanece en inglés por razones técnicas. Contiene el número de *revisión* del juego. Es un número cualquiera que el programador quiera poner aquí. Puede pensarse en él como el número de versión del juego (aunque realmente el programador puede elegir poner el número de versión como parte del **Titular**).

Serial (Antes *Serial*) El nombre de esta constante permanece en inglés por razones técnicas. Contiene el “número de serie” del juego. Debe ser una fecha de seis dígitos entre comillas dobles, aunque en teoría puede ser una cadena cualquiera de seis números. Lo normal es componer la fecha en que se ha creado esta versión concreta del juego usando, en este orden, dos cifras para el año, dos para el mes y dos para el día. Si no se define esta constante, Inform la definirá automáticamente por nosotros codificando la fecha actual (el día en que se compila el juego). Ejemplo: `Serial "020423"`.

SIN_DIRECCIONES (Antes *WITHOUT DIRECTIONS*) Si esta constante está definida antes de incluir “**EParse.h**”, entonces la librería no definirá ningún objeto dentro del objeto **brujula**. Se entiende que el jugador definirá sus propios objetos-dirección en lugar de los clásicos “norte”, “sur”, etc. No es muy habitual hacer esto.

STRICT_MODE (Antes *STRICT_MODE*) Si esta constante está definida antes de incluir la librería (no es necesario darle valor, basta definirla), entonces el juego comprenderá los verbos de depuración, y se compilará en modo *estricto*.

Titular (Antes *Headline*) Titular que se mostrará justo detrás del título. Es una descripción un poco más larga que suele incluir el nombre del autor, por ejemplo “Una historia de horror interactiva (C) 1998 Zak McKracken”.

9.2. Constantes que forman parte del lenguaje

false (Antes *false*) Esta constante representa la respuesta “falso” (por ejemplo puede usarse en comparaciones, o como valor retornado por una función). Interpretado como número sería el valor 0.

nothing (Antes *nothing*) Representa un objeto que en realidad no es ningún objeto de los del juego. Podríamos decir que es como si fuera el objeto ‘ninguno’ o ‘nada’. Se usa en comparaciones como `if (otro == nothing)`, y equivale al valor 0.

true (Antes *true*) Representa la respuesta “verdadero” (por ejemplo puede usarse en comparaciones, o como valor retornado por una función). Cuando una función retorna **true** (con la orden `return (true)` o abreviadamente `rtrue`), retorna en realidad el número 1. Sin embargo, en expresiones booleanas como `if (funciona)` cualquier valor distinto de cero de la variable global “funciona” será interpretado como **true** (y si vale cero, la expresión resultará en **false**).

9.3. Constantes definidas por la librería

CODIGO_REPARSE (Antes *REPARSE_CODE*) Código devuelto por algunas funciones para indicar al parser que debe comenzar la interpretación desde cero porque se ha modificado el buffer o la tabla de parsing.

G_FEMENINO (*Nuevo*) = 1

G_MASCULINO (*Nuevo*) = 0

G_PLURAL (*Nuevo*) = 3 El programador puede combinar estas tres constantes para asignar un valor a la propiedad *genero* de un objeto. Por ejemplo, si el objeto es femenino y plural, deberá escribir `genero G_FEMENINO + G_PLURAL`, de este modo la suma de ambas constantes produce el valor 4, que es el código asignado para el caso “femenino plural”, como se describe en la sección dedicada a la propiedad *genero*.

NULL (Antes *NULL*) Un valor usado para representar la propiedad vacía. La librería antes de llamar a una propiedad de un objeto suele comprobar que no sea igual a **NULL** (y si lo fuera, no la llama). Equivale al valor -1 (que en hexadecimal sería `$FFFF`).

9.4. Bits de formato de listas

La librería proporciona una función llamada `EscribirListaDesde` que sirve para mostrar por pantalla listas de objetos contenidos dentro de otros. Por ejemplo, cuando el jugador pide “Inventario”, se llama a esta función para que muestre todos los “contenidos” del objeto jugador. Análogamente, cuando el jugador “MIRA DENTRO DE LA CAJA”, se llama a esta función para que muestre los contenidos de la caja.

El formato de la lista que se escribirá puede ser muy variado (por ejemplo, se puede pedir que haga un salto de línea entre objeto y objeto o que no lo haga entre otras muchas opciones). El formato se define mediante una combinación de bits, a elegir entre los siguientes (nota, pueden especificarse varios a la vez simplemente sumándolos aunque algunas combinaciones de bits pueden producir extraños resultados):

BANDERAUX_BIT (Antes `WORKFLAG_BIT`) Esta máscara la usa internamente la librería. El programador de juegos no necesita conocerla. Indica que sólo se listen objetos que tengan el atributo **banderaux** activado (pero es la propia librería la que lo activa desde otra parte).

BREVE_BIT (Antes `TERSE_BIT`) Utilizar una frase en español para crear la lista, pero en un estilo más breve que el que saldría usando solamente la opción **ESPANOL_BIT**. Por ejemplo, en vez de poner “*una caja dentro de la cual hay una bola*” pondrá “*una caja (que contiene una bola)*”. Este bit debe especificarse *además* de **ESPANOL_BIT**.

DEFINIDO_BIT (Antes `DEFART_BIT`) Usar artículos definidos en lugar de indefinidos delante de cada nombre.

ESPANOL_BIT (Antes `ENGLISH_BIT`) Escribir la lista como si fuera una frase en español. Esto implica separar cada objeto del siguiente por una coma, (o “y” si es el último), añadir frases que hagan de nexos, como “dentro del cual hay...”, etc. Por ejemplo, si en el juego ponemos la orden “inventario largo”, se usará esta opción para crear la lista. Puede usarse un estilo un poco más abreviado (aunque no mucho) si además se especifica la opción **BREVE_BIT**.

HAY_BIT (Antes `ISARE_BIT`) Escribir el texto “hay” delante de la lista que se escribirá a continuación.

INDENTAR_BIT (Antes `INDENT_BIT`) Añadir espacios delante de cada objeto, para que aparezca más a la derecha si se trata de un objeto dentro de otro objeto. Lógicamente esta opción sólo tiene sentido si además se especifican **NUEVALINEA_BIT** y **RECURSIVO_BIT**.

INFOPARCIAL_BIT (Antes `PARTINV_BIT`) Escribir justo detrás de cada objeto y entre paréntesis información extra sobre su estado. Es muy similar a la opción **INFOTOTAL_BIT**, salvo algunos detalles. Por ejemplo, en vez de decir “*una caja (abierta pero vacía)*”, dirá “*una caja (que está vacía)*”.

INFOTOTAL_BIT (Antes *FULLINV_BIT*) Escribir justo detrás de cada objeto más información extra sobre el mismo entre paréntesis (por ejemplo, si está abierto o cerrado, encendido o apagado, si está vacío en caso de que sea recipiente, etc)

NUEVALINEA_BIT (Antes *NEWLINE_BIT*) Poner un retorno de carro al final de cada objeto listado (como en lo que sale al poner “Inventario breve”).

OCULTAR_BIT (Antes *CONCEAL_BIT*) Omitir de la lista todos los objetos que lleven el atributo **oculto** activado.

RECURSIVO_BIT (Antes *RECURSE_BIT*) Si uno de los objetos listados a su vez contiene objetos en su interior (y éstos pueden “verse” de acuerdo con las reglas de la librería), se listarán también los objetos contenidos. La orden “inventario” usa esta opción, tanto en modo “breve” como “largo”.

SIEMPRE_BIT (Antes *SIEMPRE_BIT*) Si algún objeto de la lista contiene a su vez otros objetos, listar también los objetos contenidos (aún cuando estos no sean “visibles” por estar el recipiente cerrado). Esta opción no suele usarse (además produce una salida un poco defectuosa creando frases como “una caja que contiene hay una bola”, debido a un bug aún no corregido).

SINARTICULO_BIT (Antes *NOARTICLE_BIT*) No usar artículos de ningún tipo delante de los nombres de los objetos.

A modo de ejemplo, diremos que el comando “Inventario breve”, lo que hace es llamar a **EscribeListaDesde**, para que liste los objetos portados por el jugador y usa las opciones: **INFOTOTAL_BIT + INDENTAR_BIT + NUEVALINEA_BIT + RECURSIVO_BIT**

El “Inventario largo”, en cambio usa las opciones **INFOTOTAL_BIT + ESPANOL_BIT + RECURSIVO_BIT**.

Cuando se muestra la descripción de una habitación, también se crea al final de la misma una lista de los objetos que contiene (para los objetos que no se han descrito ya a sí mismos a través de su propiedad *describir*, *si_abierto*, *si_cerrado*, *si_encendido* o *si_apagado*). En este caso las opciones usadas son **ESPANOL_BIT + BANDER-AUX_BIT + RECURSIVO_BIT + INFOPARCIAL_BIT + BREVE_BIT + OCULTAR_BIT**.

9.5. Códigos de error del parser

Los errores de *parsing* pueden aparecer al tratar de interpretar una orden directa (como «EXAMINA TELÉFONO») o al tratar de interpretar una orden indirecta (como «MANOLO, EXAMINA TELÉFONO»). En el primer caso la librería usará el objeto **MensajesLibreria** para que imprima un mensaje personalizado, llamando a su rutina *antes*. Este objeto debe capturar la acción *Miscelanea* y examinar la variable *ml_n* para saber el número de mensaje. Por ejemplo:

```

Object MensajesLibreria
with antes [;
    Miscelanea:
        switch(ml_n) {
            30: "No veo nada que se llame así.";
            38: "¿Pero qué dices?";
            39: "Mejor no pierdas más tiempo con eso.";
        }
];

```

Si esta rutina no existe (o no hace nada, o devuelve **false**), la librería imprimirá uno de sus mensajes estándar, que se indican a continuación junto con cada error.

En el caso de que el error haya aparecido al procesar un comando indirecto referido a otro personaje del juego, la librería no generará ningún error, sino que llamará a la rutina *ordenes* del personaje, pasándole como acción falsa *NoComprendido*. Esta rutina deberá consultar la variable **tipoerror** para saber cuál fue el error ocurrido y generar un mensaje adecuado. Si no lo hace la librería imprimirá el mensaje estándar “No hay respuesta”. Observar que la variable **tipoerror** contendrá una de las constantes que se definen a continuación, y no el número de mensaje (como ocurría con el objeto **MensajesLibreria**).

ANIMA_PE (Antes *ANIMA_PE*) = 11. Produce la respuesta “Sólo puedes hacer eso con seres animados”, que es el 37 de la acción **Miscelanea**.

ATASCADO_PE (Antes *STUCK_PE*) = 1. Produce la respuesta “No entendí esa frase”, que es la número 27 de la acción **Miscelanea**.

ESCENARIO_PE (Antes *SCENERY_PE*) = 13. Produce la respuesta estándar “Eso no es importante para el juego” que es la número 39 de la acción **Miscelanea**.

EXCEPTO_PE (Antes *EXCEPT_PE*) = 10. Produce la respuesta “Has exceptuado algo que no estaba incluido” que es la 36 de la acción **Miscelanea**

HASTAQUI_PE (Antes *UPTO_PE*) = 2. Produce la respuesta “Has puesto demasiadas palabras, sólo entendí que quieres...” y seguidamente imprime la parte del comando que ha comprendido. Este error es el número 28 de la acción falsa **Miscelanea**.

HAYPOCOS_PE (Antes *TOOFEW_PE*) = 16. Produce la respuesta “No hay tantos” cuando se intenta un comando del tipo “coge tres monedas” y no se encuentran suficientes objetos para satisfacer la petición. Es el mensaje número 42 de la acción **Miscelanea**

KKFINAL_PE (Antes *JUNKAFTER_PE*) = 15. Produce la respuesta “No entendí la última parte de la orden” que es la número 41 de la acción **Miscelanea**.

MMULTI_PE (Antes *MMULTI_PE*) = 8. Produce la respuesta “ Sólo puedes especificar objetos múltiples una vez en cada línea”, que es la número 34 de la acción **Miscelanea**.

MULTI_PE (Antes *MULTI_PE*) = 7. Produce la respuesta “No puedes especificar objetos múltiples con ese verbo”, que es la número 33 de la acción **Miscelanea**

MUYPOCO_PE (Antes *TOOLIT_PE*) = 5. Produce la respuesta “¡Pareces haber dicho muy poca cosa!”, que es la número 31 de la acción **Miscelanea**.

NADA_PE (Antes *NOTHING_PE*) = 17. Produce una de las respuestas “¡No hay que hacer nada!” (número 43 de la acción **Miscelanea**) o “No hay nada inmediatamente disponible” (número 44 de la acción **Miscelanea**). Está relacionado con la especificación de objetos múltiples.

NOTIENES_PE (Antes *NOTHELD_PE*) = 6. Produce la respuesta “¡No tienes nada de eso!”, que es la número 32 de la acción **Miscelanea**.

NOVEO_PE (Antes *CANTSEE_PE*) = 4. Produce la respuesta “No veo eso que dices”, que es la número 30 de la acción **Miscelanea**. Esta respuesta es la típica dada cuando el jugador ha escrito mal el nombre de un objeto (por ejemplo: “MIRA TELÉFONO”). El jugador puede corregir a continuación la palabra que había escrito mal poniendo “eepa teléfono”).

NUMERO_PE (Antes *NUMBER_PE*) = 3. Produce la respuesta “No comprendí ese número”, que es la número 29 de la acción **Miscelanea**. Esta respuesta aparece cuando, de acuerdo con la gramática el jugador debería haber escrito un número como parte del comando, pero no se ha podido encontrar o interpretar correctamente ese número.

PREGUNTAAMBITO_PE (Antes *ASKSCOPE_PE*) = 18. Este error no produce mensaje, sino que causa un intento por parte de la librería de refinar el problema y encontrar otro mensaje de error mejor.

PRONOM_PE (Antes *VAGUE_PE*) = 9. Produce un mensaje del tipo “No estoy seguro de a qué se refiere lo”, que es el número 35 de la acción **Miscelanea**. Puede aparecer en respuesta a un comando como “EXAMÍNALO” cuando el pronombre “-lo” no había sido asignado previamente.

VERBO_PE (Antes *VERB_PE*) = 12. Produce la respuesta “No conozco ese verbo”, que es la número 38 de la acción *Miscelanea*. Aparece cuando la primera palabra del comando del jugador no se ha podido interpretar como un verbo ni como el nombre de una de las direcciones de la **brujula**.

YANOPRON_PE (Antes *IT_GONE*) = 14. Produce un mensaje del tipo “Ahora mismo no puedes ver lo que representa el pronombre lo”(el cofre)”, que es el número 40 de la acción *Miscelanea*. Aparece ante órdenes como “EXAMÍNALO”, cuando el pronombre “-lo” se refiere a un objeto que ya no está al alcance del jugador.

9.6. Definiciones del idioma

En la librería original, estas constantes contenían ciertas palabras que el parser necesitaba. Se definían como constantes en lugar de usar directamente las palabras en cuestión para hacer de este modo más configurable la librería. En la librería española ya no tienen mucho sentido, pero se mantienen por compatibilidad.

El programador de juegos no necesitará conocer ni cambiar estas constantes. Sólo se listan como referencia.

ANULAR1__WD (Antes *UNDO1__WD*) La palabra que representa la acción “anular” (definida en el fichero dependiente del idioma). Valor = ‘undo’

ANULAR2__WD (Antes *UNDO2__WD*) Otra palabra que represente la acción “anular” (definida en el fichero dependiente del idioma). Valor = ‘deshacer’

ANULAR3__WD (Antes *UNDO3__WD*) Otra palabra que represente la acción “anular” (definida en el fichero dependiente del idioma). Valor = ‘anular’

CURIOSIDADES__WD (Antes *AMUSING__WD*) La palabra que representa la opción “ver otras curiosidades” que se ofrece al jugador cuando ha completado el juego. Valor = ‘curiosidades’

DE1__WD (Antes *OF1__WD*) Palabra que representa a la conjunción “de”. En la librería española la hemos definido a un valor “inescribible”, para que el parser nunca pueda encontrarla entre lo que ha escrito el jugador, ya que en la librería española esta preposición la maneja la rutina **InterpretarNombre** mejor que el parser original. Por esa razón tiene el valor = ‘.de’. El punto como parte de la palabra la convierte en “inescribible” por el jugador (ya que aunque el jugador la escribiera así, la máquina Z la separaría en dos palabras: el punto y el ‘de’).

DE2__WD (Antes *OF2__WD*) Otro sinónimo de “de”. También tiene valor = ‘.de’

DE3__WD (Antes *OF3__WD*) Otro sinónimo de “de”. También tiene valor = ‘.de’

DE4__WD (Antes *OF4__WD*) Otro sinónimo de “de”. También tiene valor = ‘.de’

DESPUES1__WD (Antes *THEN1__WD*) Palabra que representa la conexión “y después” para el caso de que el jugador quiera conectar así varias acciones en un mismo comando. Tiene valor = ‘ydespues’. Lo normal será que el jugador ponga “y después” en dos palabras separadas, pero esto es detectado por la rutina **EspanolAInformes** (que es la que hace transformaciones en lo que el jugador haya escrito antes de pasársela al parser), y allí se juntan estas palabras en una sola, puesto que la conjunción “y” por separado tiene otro significado para el parser.

DESPUES11__WD (*Nuevo*) Esta constante tiene el valor que **EspanolAInformes** buscará detrás de una “y” para detectar cuándo debe juntarlas en una sola palabra de valor **DESPUES1__WD**. Valor = ‘despues’

DESPUES2__WD (Antes *THEN2__WD*) Otro sinónimo para “ydespues”. Valor = ‘yluego’

DESPUES21__WD (*Nuevo*) Análogo a **DESPUES11__WD**. Valor = ‘luego’

DESPUES3__WD (Antes *THEN3__WD*) Otro sinónimo para “ydespues”. Valor = ‘yentonces’

DESPUES31__WD (*Nuevo*) Análogo a **DESPUES11__WD**. Valor = ‘entonces’

ELCUAL__TX (Antes *WHICH__TX*) Nexo usado por la librería para construir mensajes al crear listas de objetos. Valor = "que "

ESASC__TX (Antes *THOSET__TX*) Texto usado para referirse a un grupo de cosas desconocidas (impreso en algunos mensajes de error). Valor = "esas cosas"

ESO__TX (Antes *THAT__TX*) Texto usado para referirse a una cosa desconocida (impreso en algunos mensajes de error). Valor = "eso"

HAY__TX (Antes *IS__TX*) Texto usado delante del listado de una lista cuando la opción **HAY_BIT** está activada y la lista contiene sólo un elemento. Valor = "hay"

HAYP__TX (Antes *ARE__TX*) Texto usado delante del listado de una lista cuando la opción **HAY_BIT** está activada y la lista contiene más de un elemento. Valor = "hay"

HAY2__TX (Antes *IS2__TX*) Texto usado como cabecera antes de listar lo que hay en un recipiente (cuando sólo hay un objeto dentro). Valor = "hay "

HAYP2__TX (Antes *ARE2__TX*) Texto usado como cabecera antes de listar lo que hay en un recipiente (cuando hay más de un objeto dentro). Valor = "hay "

HORA__TX (Antes *TIME__TX*) Texto usado como encabezado de la hora (cuando la librería necesite mostrar la hora, lo cual puede ocurrir en algunos casos en la barra de estado). Valor = "Hora: "

JUGADAS__TX (Antes *MOVES__TX*) Texto usado delante del contador de jugadas (en la barra de estado). Valor = "Movim.: "

NADA__TX (Antes *NOTHING__TX*) Texto que sirve de nombre para el objeto nulo. Valor = "nada"

NO1__WD (Antes *NO1__WD*) Palabra de diccionario que el representa la respuesta “no” a una pregunta del tipo “¿Estás seguro?”. Valor = ‘n//’ (recordar que las palabras de diccionario compuestas por una sola letra, como este caso, deben llevar doble barra tras la letra. De lo contrario Inform no las introducirá en el diccionario, sino que las almacenará como una constante de tipo carácter).

NO2__WD (Antes *NO2__WD*) Otro sinónimo de la anterior. Valor = ‘no’

NO3__WD (Antes *NO3__WD*) Un sinónimo más. Valor = 'no' (en español ya no hay más formas de decir que no).

NOPUEDESIR__TX (Antes *CANTGO__TX*) Texto a imprimir por defecto cuando el jugador intenta ir por una salida que no está definida en esa localización. Valor = "No puedes ir por ahí."

OOPS1__WD (Antes *OOPS1__WD*) Palabra de diccionario que el parser comprenderá como "oops" (este es un comando especial para el parser que permite corregir una palabra no comprendida del comando anterior. Por ejemplo si el jugador pone "SACA LA BOLA ROJA DE LA CAHA" la librería dirá algo como "No veo eso que dices", entonces el jugador puede rectificar poniendo "OOPS CAJA" y el parser cambiará la palabra "CAHA" que no había comprendido por "CAJA". Valor = 'oops')

OOPS2__WD (Antes *OOPS2__WD*) Otro sinónimo del anterior. Valor = 'epa'

OOPS3__WD (Antes *OOPS3__WD*) Otro sinónimo del anterior. Valor = 'eepa'

OSCURIDAD__TX (Antes *DARKNESS__TX*) Cadena de caracteres que sirve de nombre al objeto **laoscuridad** (este es el objeto dentro del cual la librería mueve al jugador cada vez que se queda a oscuras). Valor = "Oscuridad"

OTRAVEZ1__WD (Antes *AGAIN1__WD*) Palabra de diccionario que el parser entenderá como una petición de "repetir el comando anterior". Valor = 'repetir'. No es un verbo normal (no está en la gramática) sino que es directamente detectado y manejado por el parser.

OTRAVEZ2__WD (Antes *AGAIN2__WD*) Otro sinónimo para el anterior. Valor = 're'

OTRAVEZ3__WD (Antes *AGAIN3__WD*) Otro sinónimo para el anterior. Valor = 'g/' (por compatibilidad con juegos en inglés que usan "g" como abreviatura de "again").

OTRO1__WD (Antes *OTHER1__WD*) Palabra comprendida por el parser como artículo indefinido para comandos del tipo "Coge otro objeto". Valor = 'otro'

OTRO2__WD (Antes *OTHER2__WD*) Sinónimo para el anterior. Valor = 'otra'

OTRO3__WD (Antes *OTHER3__WD*) Otro sinónimo para el anterior. Valor = 'otro'

O__TX (Antes *OR__TX*) Texto a imprimir para separar varias alternativas, usado por la librería para generar mensajes del tipo "¿cuál exactamente, la caja roja o la caja verde?". Valor = .° "

PREVIOYO__TX (Antes *FORMER__TX*) Cadena de texto a imprimir cuando la librería necesite referirse al objeto **objjugador** (que es el objeto que normalmente representa al jugador) cuando el jugador ya no es ese objeto (por ejemplo, en juegos en los que el jugador pueda pasar su "espíritu" de un ser a otro. Valor = "tu antiguo yo ". Hay que decir que raro es el juego que usa esta posibilidad.

PUNTUACION1__WD (Antes *FULLSCORE1__WD*) Palabra de diccionario que representa la opción “Ver la puntuación” que se ofrece al final del juego. Valor = ‘puntuacion’.

PUNTUACION2__WD (Antes *FULLSCORE__WD*) Un sinónimo para la anterior. Valor = ‘punt’.

PUNTUACION__TX (Antes *SCORE__TX*) Cadena de caracteres a imprimir en la barra de estado delante de la puntuación actual. Valor = "Punt . : "

QKEY1__KY (Antes *QKEY1__KY*) Tecla que representa la opción “abandonar” cuando se está dentro de un menú, para volver al juego. Valor = ‘Q’ (es un carácter, no una palabra de diccionario).

QKEY2__KY (Antes *QKEY2__KY*) Lo mismo que el anterior, pero cuando se está dentro de un submenú para salir al menú anterior. Valor = ‘Q’.

QKEY1__TX (Antes *QKEY1__TX*) Texto a mostrar en un menú para la opción de salir del mismo. Valor = "Q=Volver al juego"

QKEY2__TX (Antes *QKEY2__TX*) Texto a mostrar en un submenú para salir del mismo. Valor = "Q = Menú anterior"

QUIEN__TX (Antes *WHOM__TX*) Nexo utilizado en la construcción de listas de objetos, para mostrar los contenidos de un objeto animado. Valor = "que "

RECUPERAR__WD (Antes *RESTORE__WD*) Palabra de diccionario que representa la opción “recuperar un juego salvado” entre las que se ofrecen al final del juego. Valor = ‘recuperar’

REINICIAR__WD (Antes *RESTART__WD*) Palabra de diccionario que representa la opción “reiniciar el juego” entre las que se ofrecen al final del juego. Valor = ‘reiniciar’

RKEY__TX (Antes *RKEY__TX*) Texto a mostrar en la barra de estado de un menú como ayuda para indicar qué tecla debe pulsarse para seleccionar una opción de menú. Valor = "INTRO = leer este tema"

SALVO1__WD (Antes *BUT1__WD*) Palabra de diccionario que el parser entenderá como “excepto” en las frases del tipo “Coge todas las monedas excepto la herrumbrosa” y similares. Valor = ‘excepto’

SALVO2__WD (Antes *BUT2__WD*) Otro sinónimo para la anterior. Valor = ‘menos’

SALVO3__WD (Antes *BUT2__WD*) Un sinónimo más para la anterior. Valor = ‘salvo’

SII__WD (Antes *YES1__WD*) Palabra de diccionario que representa la respuesta “sí” en las preguntas de tipo “¿Estás seguro?”. Valor = ‘s / /’. Observar la doble barra (que indica que es una palabra de diccionario que consta de un solo carácter).

SI2__WD (Antes *YES2__WD*) Un sinónimo para la anterior. Valor = ' si '

SI3__WD (Antes *YES3__WD*) Otro sinónimo para la anterior. Valor = ' sí '

TECLAANT1__KY (Antes *PKEY1__KY*) Tecla que pasa a la opción anterior en un menú. Valor = ' P ' (es un carácter, no una palabra de diccionario, por eso no lleva doble barra).

TECLAANT2__KY (Antes *PKEY2__KY*) Lo mismo que la anterior. Valor = ' p '

TECLAANT__TX (Antes *PKEY__TX*) Texto de ayuda a mostrar en la barra de estado de los menús, indicando qué tecla debe usarse para elegir la opción previa en un menú. Valor = "P=Previo"

TECLASIG1__KY (Antes *NKEY1__KY*) Tecla que pasa a la opción siguiente en un menú. Valor = ' S ' (es un carácter, no una palabra de diccionario, por eso no lleva doble barra).

TECLASIG2__KY (Antes *NKEY2__KY*) Lo mismo que la anterior. Valor = ' s '

TECLASIG__TX (Antes *NKEY__TX*) Texto de ayuda a mostrar en la barra de estado de los menús, indicando qué tecla debe usarse para elegir la opción siguiente en un menú. Valor = "S = Siguiente"

TERMINAR1__WD (Antes *QUIT1__WD*) Palabra de diccionario que representa la opción “terminar” entre las que aparecen al final del juego. Valor = ' q / / '

TERMINAR2__WD (Antes *QUIT2__WD*) Un sinónimo para la anterior. Valor = ' terminar '

TODO1__WD (Antes *ALL1__WD*) Palabra de diccionario que será interpretada por el parser como una referencia a “todos los objetos”. Valor = ' todos '

TODO2__WD (Antes *ALL2__WD*) Un sinónimo para la anterior. Valor = ' todas '

TODO3__WD (Antes *ALL3__WD*) Otro sinónimo para la anterior. Valor = ' todo '

TODO4__WD (Antes *ALL4__WD*) Otro sinónimo para la anterior. Valor = ' ambos '

TODO5__WD (Antes *ALL5__WD*) Un sinónimo más para la anterior. Valor = ' ambas '

TUMISMO__TX (Antes *YOURSELF__TX*) Cadena de caracteres a mostrar cuando la librería necesite imprimir el nombre del jugador. Valor = "a ti mismo".

Y1__WD (Antes *AND1__WD*) Palabra de diccionario comprendida por el parser como la conjunción “y”. Valor = ' y / / '

Y2__WD (Antes *AND2__WD*) Sinónimo para la anterior. Valor = ' y / / '

Y3__WD (Antes *AND3__WD*) Otro sinónimo para la anterior. Valor = ' e / / '

YO1__WD (Antes *ME1__WD*) Palabra de diccionario comprendida por el parser como una referencia al objeto jugador. Valor = ' -me '. Nota, la función *EspanolAlnformes* detecta si el verbo usado termina en “me” y entonces lo separa, por lo que si el jugador escribe “MIRAME”, el parser recibe “MIRA -ME”, de ahí esta definición.

YO2__WD (Antes *ME2__WD*) Otro sinónimo para la anterior. Valor = ' -te ' (por si el jugador pone “MIRATE”).

YO3__WD (Antes *ME3__WD*) Un sinónimo más para el anterior. Valor = ' ti ' (para el raro caso de que ponga “MIRAR A TI”).

Y__TX (Antes *AND__TX*) Cadena de caracteres para usar como conjunción copulativa en las listas de elementos impresas por la librería (como en “puedes ver un manojito de llaves y una linterna”). Valor = " y "

9.7. Constantes que representan razones

Cuando el parser llama a rutinas de usuario para averiguar si un objeto o no está en la lista de cosas a las que el jugador puede referirse (su “alcance”) previamente almacena en la variable **razon_alcance** una de las siguientes constantes. De este modo la rutina de usuario puede responder de forma diferente según cuál haya sido la razón de la llamada. Esto entra dentro de los conceptos de programación muy avanzada.

RAZON_BUCLEALCANCE (Antes *LOOPOVERSCOPE_REASON*) = 5

RAZON_CADA_TURN (Antes *EACH_TURN_REASON*) = 2

RAZON_HABLAR (Antes *TALKING_REASON*) = 1

RAZON_PARSING (Antes *PARSING_REASON*) = 0

RAZON_REACCIONAR_ANTES (Antes *REACT_BEFORE_REASON*) = 3

RAZON_REACCIONAR_DESPUES (Antes *REACT_AFTER_REASON*) = 4

RAZON_TESTALCANCE (Antes *TESTSCOPE_REASON*) = 6

9.8. Otras constantes oscuras

Las siguientes constantes son definidas por la librería y el parser para uso interno. No daremos muchos detalles sobre ellas, pero las listamos aquí para que conste su equivalencia con los originales en inglés.

APAGADO_BIT (Antes *UNLIT_BIT*) Usado por la función **Adjudicar**

ENCENDIDO_BIT (Antes *LIT_BIT*) Usado por la función **Adjudicar**

ESO_BIT (Antes *THAT_BIT*) Usado por la función **Adjudicar**

MI_BIT (Antes *MY_BIT*) Usado por la función **Adjudicar**

OTRO_BIT (Antes *OTHER_BIT*) Usado por la función **Adjudicar**

PLURAL_BIT (Antes *PLURAL_BIT*) Usado por la función **Adjudicar**

CasosLenguaje (Antes *LanguageCases*) Usado en algunos idiomas.

Crear__IN (Antes *Make__PN*) Si el parser debe crear una rutina InterpretarNombre por defecto.

Crear__IR (Antes *Make__PR*) Si el parser debe crear una rutina ImprimirRango por defecto.

DEFINIDO_PK (Antes *DEFART_PK*) Usado para construir tablas de artículos, posesivos, etc.

Grammar__Version (Antes *Grammar__Version*) Esta constante debe estar definida en algún lugar de la librería (y con su nombre en inglés) y debe tener un valor mayor o igual que 2. De lo contrario Inform rehusará compilarla.

IdiomaContracciones (Antes *LanguageContractionForms*) Cuántas formas diferentes de contracción tiene el idioma que “habla” el juego. El español no tiene contracciones de éstas (se refiere a contraer ciertos artículos delante de algunos nombres, como en el francés “l’enfant”).

IdiomaGeneroAnimado (Antes *LanguageAnimateGender*) Cuál es el género por defecto para los seres vivos (en español está definido como masculino).

IdiomaGeneroInanimado (Antes *LanguageInanimateGender*) Cuál es el género por defecto para los objetos no animados (en español está definido como masculino).

NumSerieLib (Antes *LibSerial*) Número de serie de la librería (la fecha en que fue publicada).

PATRON_NULO (Antes *NULL_PATTERN*) Es el valor asignado a un elemento de la tabla de parsing cuando no ha coincidido con nada conocido. Tiene el valor \$FFFF.

RevisionLib (Antes *LibRelease*) Número de revisión de la librería. Contiene la cadena "6/10E" (la E de “Español” la distingue de la librería original).

RPG_CRIATURA (Antes *GPR_CREATURE*) = \$ff06. Valor que puede ser retornado como respuesta por una rutina de parsing genérica.

RPG_FALLO (Antes *GPR_FAIL*) = -1. Valor que puede ser retornado como respuesta por una rutina de parsing genérica.

RPG_MULTI (Antes *GPR_MULTI*) = \$ff02. Valor que puede ser retornado como respuesta por una rutina de parsing genérica.

RPG_MULTIDENTRO (Antes *GPR_MULTIINSIDE*) = \$ff05. Valor que puede ser retornado como respuesta por una rutina de parsing genérica.

RPG_MULTIEXCEPTO (Antes *GPR_MULTIEXCEPT*) = \$ff04. Valor que puede ser retornado como respuesta por una rutina de parsing genérica.

RPG_MULTIPOSEIDO (Antes *GPR_MULTIHELD*) = \$ff03. Valor que puede ser retornado como respuesta por una rutina de parsing genérica.

RPG_NOMBRE (Antes *GPR_NOUN*) = \$ff00. Valor que puede ser retornado como respuesta por una rutina de parsing genérica.

RPG_NUMERO (Antes *GPR_NUMBER*) = 1. Valor que puede ser retornado como respuesta por una rutina de parsing genérica.

RPG_POSEIDO (Antes *GPR_HELD*) = \$ff01. Valor que puede ser retornado como respuesta por una rutina de parsing genérica.

RPG_PREPOSICION (Antes *GPR_PREPOSITION*) = 0. Valor que puede ser retornado como respuesta por una rutina de parsing genérica.

RPG_REPARSE (Antes *GPR_REPARSE*) = **CODIGO_REPARSE**. Valor que puede ser retornado como respuesta por una rutina de parsing genérica.

TAMANIO_LISTA_ENCAJAN (Antes *MATCH_LIST_SIZE*) = 128. Tamaño máximo de la lista que contiene todos los objetos que podrían encajar con lo escrito por el jugado (en particular, esto implica que cuando el jugador pone “coge todo”, no puede haber más de 128 objetos “cogibles”).

TOKEN_CRIATURA (Antes *CREATURE_TOKEN*) = 6. Usado por el parser.

TOKEN_ESPECIAL (Antes *SPECIAL_TOKEN*) = 7. Usado por el parser.

TOKEN_FINAL (Antes *ENDIT_TOKEN*) = 15. Usado por el parser.

TOKEN_MULTI (Antes *MULTI_TOKEN*) = 2. Usado por el parser.

TOKEN_MULTIDENTRO (Antes *MULTIINSIDE_TOKEN*) = 5. Usado por el parser.

TOKEN_MULTIEXCEPTO (Antes *MULTIEXCEPT_TOKEN*) = 4. Usado por el parser.

TOKEN_MULTIPOSEIDO (Antes *MULTIHELD_TOKEN*) = 3. Usado por el parser.

TOKEN_NOMBRE (Antes *NOUN_TOKEN*) = 0. Usado por el parser.

TOKEN_NUMERO (Antes *NUMBER_TOKEN*) = 8. Usado por el parser.

TOKEN_POSEIDO (Antes *HELD_TOKEN*) = 1. Usado por el parser.

TOKEN_TEMA (Antes *TOPIC_TOKEN*) = 9. Usado por el parser.

TT_ALCANCE (Antes *SCOPE_TT*) = 5. Tipo de elemento de la gramática que representa una función de *scope*.

TT_ELEMENTAL (Antes *ELEMENTARY_TT*) = 1. Tipo de elemento básico de la gramática. Puede ser de uno de los subtipos definidos en las constantes **TOKEN_***.

TT_FILTRO_ATRIB (Antes *ATTR_FILTER_TT*) = 4. Tipo de elemento de la gramática que representa un atributo (y por tanto sólo admitirá objetos que tengan ese atributo)

TT_FILTRO_RUTINA (Antes *ROUTINE_FILTER_TT*) = 3. Tipo de elemento de la gramática que representa un filtro de rutina (o sea, que sólo dejará pasar los objetos para los cuales la rutina devuelva **true**).

TT_ILEGAL (Antes *ILLEGAL_TT*) = 0. Tipo de elemento de la gramática inválido.

TT_PREPOSICION (Antes *PREPOSITION_TT*) = 2. Tipo de elemento de la gramática que representa una preposición, por ejemplo ‘sobre’ (en realidad cualquier palabra entre comillas es interpretada como preposición en este contexto).

TT_RPG (Antes *GPR_TT*) = 6. Tipo de elemento de la gramática que representa una rutina de parsing general. Esta rutina deberá devolver una de las constantes **RPG_*** para indicar al parser de qué tipo es el token procesado.

VersionIdioma (Antes *LanguageVersion*) Número de versión de las librerías correspondientes al idioma o dialecto concreto que se esté usando. En la versión española InformATE!, esta constante está definida en el fichero *Espanol.h*, y contiene el texto: "Librería Española InformATE! 020804".

Capítulo 10

Índice de identificadores

En los índices, junto a cada identificador se indica entre paréntesis una de las siguientes aclaraciones, que indican el tipo de identificador: (Acc) acción, (Atr) atributo, (Cons) constante, (Fich) nombre de fichero, (Prop) propiedad, (Rut) rutina o punto de entrada, (Var) variable, (Obj) objeto.

10.1. Identificadores de InformATE!

En este índice aparecen todos los identificadores presentes en la librería española InformATE!, así como los números de página en los que hay información interesante sobre cada uno de ellos. Un número de página en **negrita** indica que en esa página es donde se comenta en profundidad el identificador correspondiente.

Símbolos	
(MCoge) (Rut)	80
(MMCoge) (Rut)	80
(_El) (Rut)	79
(_nombre_) (Rut)	80
(al) (Rut)	79
(coge) (Rut)	79, 80
(del) (Rut)	79
(e) (Rut)	79
(el) (Rut)	79
(es) (Rut)	80
(esta) (Rut)	80
(lo) (Rut)	80
(n) (Rut)	80
(numero) (Rut)	80
(o) (Rut)	80
(s) (Rut)	80
(un) (Rut)	80
A	
abajo (Prop)	17, 21, 77
abierta (Atr)	11
abierto (Atr) ..	11, 13–15, 19, 22, 32, 33, 48, 49, 54, 62, 70
abrible (Atr)	11, 13–15, 19, 26, 32, 33, 48, 49
Abrir (Acc)	11, 12, 18, 48
accion (Var)	24, 25, 28, 29, 35, 37, 64, 67, 67
accion_invertida (Var)	70
accion_parser (Var)	64, 71
accion_que_seria (Var)	71
accion_recibir (Var)	55, 59, 71
Acciones.h (Fich)	9, 60
accionm.h (Fich)	9
ActivarAcciones (Acc)	62
ActivarAcentos (Acc)	62
ActivarComandos (Acc)	62
ActivarNotificacion (Acc)	39
ActivarRelojos (Acc)	62
ActivarRutinas (Acc)	63
ActivarTranscripcion (Acc)	39
ActivarTraza (Acc)	63
actor (Var)	67, 67, 87
ActualizarEstatus (Rut)	90
ActualizarPronombre (Rut)	81

ActualizarTiempoReal (Rut).....	69	bandera_debug (Var).....	71
adentro (Prop).....	17, 21, 52, 77	bandera_demasiados (Var).....	71
adjetivos (Prop).....	5, 14, 18, 88, 91	bandera_deshacer (Var).....	71
ADMITIR_COMANDO_SALIDAS (Cons).....	32, 60, 95	bandera_guapo (Var).....	71
ADMITIR_INFINITIVOS (Cons).....	95	bandera_imprime_jugador (Var).....	71
afuera (Prop).....	18, 21, 61, 77	bandera_paa (Var).....	71
Agitar (Acc).....	41	bandera_puesto_en (Var).....	71
AjustarLuz (Rut).....	90	banderafin (Var).....	54, 67, 89
al_e (Prop).....	18, 21, 77	banderaluz (Var).....	71
al_n (Prop).....	18, 21, 22, 54, 60, 77	banderamulti (Var).....	71
al_ne (Prop).....	18, 21, 77	banderaux (Atr).....	11
al_no (Prop).....	18, 21, 77	BANDERAUX_BIT (Cons).....	99
al_o (Prop).....	18, 21, 77	Beber (Acc).....	41
al_s (Prop).....	18, 21, 22, 77	Besar (Acc).....	35, 41
al_se (Prop).....	18, 21, 77	best_tipoerror (Var).....	71
al_so (Prop).....	18, 21, 77	BorrarDescriptores (Rut).....	90
AlAlcance (Rut).....	87	BREVE_BIT (Cons).....	99
Alcance (Acc).....	63	brujula (Obj).....	22, 53, 60, 77, 78, 97, 102
amodo_noposeido (Var).....	71	BucleAlcance (Rut).....	82
analizar_input (Prop).....	78	buffer (Var).....	71, 83, 87
AnalizarToken (Rut).....	90	buffer2 (Var).....	71
ancho_elemento (Var).....	71	buffer3 (Var).....	71
AniadirMulti (Rut).....	90	bufferaux (Var).....	71
anidacion_menu (Var).....	71	BuscarEn (Acc).....	37, 49, 52
ANIMA_PE (Cons).....	101	BuscarEnAlcance (Rut).....	90
animado (Atr).....	11, 11, 24, 28, 44, 48	BuscarEnDiccionario (Rut).....	71, 82
AnotarLlegada (Rut).....	90	BuscarEntreVerbosIrregulares (Rut).....	90
AnotarObjetosObtenidos (Rut).....	90		
AntepasadoComun (Rut).....	90	C	
antes (Prop) ..	18, 19, 21, 31, 32, 35, 37, 38, 41–47, 49–51, 53–55, 59, 61, 63, 65, 67, 77, 78, 85, 89, 100	cada_turno (Prop).....	20
AntesDelParsing (Rut).....	87	CadenaDePreposiciones (Rut).....	90
ANULAR1__WD (Cons).....	103	CambiarDefecto (Rut).....	82
ANULAR2__WD (Cons).....	103	CambiarJugador (Rut).....	68, 78, 82
ANULAR3__WD (Cons).....	103	CaminarAOscuras (Rut).....	54, 87
Anuncio (Rut).....	90	Cantar (Acc).....	42
APAGADO_BIT (Cons).....	108	cantidad (Prop).....	20, 82
Apagar (Acc).....	12, 48, 53	capacidad (Prop).....	20, 55, 59
ArrancarDaemon (Rut).....	20, 81	caso_nombre_corto (Var).....	71
ArrancarReloj (Rut).....	34, 82	CasosLenguaje (Cons).....	109
arriba (Prop).....	19, 21, 77	CDefArt (Rut).....	82, 82
articulo (Prop).....	19, 19	Cerrar (Acc).....	11, 49
articulos (Prop).....	13–15, 19, 79, 80	cerrojo (Atr).....	11, 12, 15, 20, 51, 59
Atacar (Acc).....	35, 41, 41, 42	cerrojoechado (Atr) ..	12, 12, 15, 19, 48, 51, 52, 59, 60, 70
Atar (Acc).....	41	child (Rut).....	80
ATASCADO_PE (Cons).....	101	children (Rut).....	81
ausente (Atr).....	11, 11	CientosEspanol (Rut).....	90
AveriguarPreposicion (Rut).....	90	codigo_razon (Var).....	58, 71
AveriguarPrimeraPreposicion (Rut).....	90	CODIGO_REPARSE (Cons).....	98
aviso_avanzar (Var).....	71	Coger (Acc).....	31, 39, 49, 49, 55, 60, 62
		Columpiar (Acc).....	42
		Comer (Acc).....	12, 50, 50
		comestible (Atr).....	12, 50
		CompararSinSigno (Rut).....	82
		Comprar (Acc).....	42
B			
Bajar (Acc).....	48		

con_llave (Prop)	12, 20, 51, 59	Despertarse (Acc)	42
conmutable (Atr)	12, 27, 33, 34, 46, 48, 52, 53	despues (Prop)	21, 38, 48, 50, 56, 60, 67
Conseguido (Rut)	82, 88	despues (Rut)	48, 52, 59, 60, 62, 70, 86
Consultar (Acc)	42, 42	DESPUES11_WD (Cons)	103
consultar_desde (Var)	29, 46, 47, 71	DESPUES1_WD (Cons)	103
consultar_num_palabras (Var)	29, 46, 47, 71	DESPUES21_WD (Cons)	104
contadorp (Var)	71	DESPUES2_WD (Cons)	104
contadorp2 (Var)	71	DESPUES31_WD (Cons)	104
ContieneIndirectamente (Rut)	90	DESPUES3_WD (Cons)	104
CopiarBuffer (Rut)	90	Desvestir (Acc)	51, 55, 59
Cortar (Acc)	42, 42	Dialecto (Acc)	39
Crear__IN (Cons)	109	dialecto_sudamericano (Var)	68
Crear__IR (Cons)	109	DialectoCast (Acc)	39
Curiosidades (Rut)	87, 95, 96	DialectoSud (Acc)	39
CURIOSIDADES__WD (Cons)	103	DibujarLineaEstado (Rut)	83, 83
D		DigitoEspanol (Rut)	91
daemon (Prop)	20, 81, 85	DirDicc__Num (Rut)	91
Dar (Acc)	35, 50	direcc_puerta (Prop)	15, 22, 22, 77
dat1 (Var)	71	DireccionDePalabra (Rut)	83
dat2 (Var)	72	DoMenu (Rut)	83
DE1__WD (Cons)	103	DominioNombre (Rut)	83
DE2__WD (Cons)	103	Dormir (Acc)	42
DE3__WD (Cons)	103	E	
DE4__WD (Cons)	103	EcharCerrojo (Acc)	51
DeboBajarRecurso (Rut)	90	eepa_desde (Var)	72
DEBUG (Cons)	38, 95	eepa_guardado (Var)	72
DecirQueHaySobre (Rut)	90	EfectuarAccionesAlcance (Rut)	91
deducedesde (Var)	72	EjecutarRutinas (Rut)	91
DefArt (Rut)	83, 83	EjecutarVida (Rut)	91
DEFINIDO_BIT (Cons)	99	ELCUAL__TX (Cons)	104
DEFINIDO_PK (Cons)	109	ElegirObjetos (Rut)	88
Dejar (Acc)	51, 51, 55, 58, 62	elemento_menu (Var)	72
DejarSalir (Acc)	50, 63, 63	ElementoCualquiera (Rut)	83
DentroDelAlcance (Rut)	83	EligeObjetos (Rut)	91
DentroDelAlcance_O (Rut)	90	EliminarDuplicados (Rut)	91
DepurarAccion (Rut)	90	ElMismo (Acc)	64
DepurarAtributo (Rut)	91	elsiguiente (Var)	72
DepurarLineaGramatica (Rut)	91	Empujar (Acc)	42, 44
DepurarParametro (Rut)	91	EmpujarDir (Acc)	43, 43, 85
DepurarToken (Rut)	91	encajado_desde (Var)	72
DesactivarAcciones (Acc)	63	Encender (Acc)	12, 41, 46, 52, 52, 53
DesactivarAcentos (Acc)	63	encendida (Atr)	12
DesactivarComandos (Acc)	63	encendido (Atr)	12, 33-35, 48, 52
DesactivarNotificacion (Acc)	39	ENCENDIDO_BIT (Cons)	109
DesactivarReloj (Acc)	63	EnglishNumber (Rut)	83, 83, 91
DesactivarRutinas (Acc)	63	enlazpa.h (Fich)	9, 10
DesactivarTranscripcion (Acc)	39	enlazlv.h (Fich)	10
DesactivarTraza (Acc)	63	entable (Atr)	12, 21, 56
describir (Prop)	13, 20, 25, 57, 58, 85, 100	Entrar (Acc)	12, 52
descripcion (Prop)	20, 21, 52, 57	EParser.h (Fich)	9, 10, 32, 69, 77, 83
descripcion_dentro (Prop)	21, 58	Eparserm.h (Fich)	9
Descriptores (Rut)	91	error_alcance (Var)	72
DesempaquetarLineaGramatica (Rut)	91	ErrorDeEjecucion (Rut)	91
DespertarOtro (Acc)	35, 42	ErrorParser (Rut)	88

ESASC__TX (Cons)	104	Hablar (Acc)	35, 44
escenario (Atr)	12, 44, 48, 50, 57	HacerEncaje (Rut)	91
ESCENARIO_PE (Cons)	101	HalladoPlural (Acc)	64
EscribeAntesElemento (Rut)	91	HASTAQUI_PE (Cons)	101
EscribeListaR (Rut)	91	HAY2__TX (Cons)	104
EscribeTrasElemento (Rut)	91	HAY__TX (Cons)	104
EscribirListaDesde (Rut)	84, 91, 99	HAY_BIT (Cons)	99
Escuchar (Acc)	31, 43	HAY_CURIOSIDADES (Cons)	87, 95
ESO__TX (Cons)	104	HAY_TAREAS (Cons)	70, 88, 96, 97
ESO_BIT (Cons)	109	HayLuz (Rut)	84
Espanol.h (Fich)	9, 10, 111	HAYP2__TX (Cons)	104
ESPAÑOL_BIT (Cons)	99	HAYP__TX (Cons)	104
EspanolAInformes (Rut)	70, 91, 108	HAYPOCOS_PE (Cons)	101
Esperar (Acc)	43	herobj (Var)	72
esta_en (Prop)	11, 22	himobj (Var)	72
estadio_alcance (Var)	72	Historia (Cons)	96
estatica (Atr)	12	HORA__TX (Cons)	104
estatico (Atr)	12, 44, 48, 50	hora_freq (Var)	72
estilo_ac (Var)	72	hora_incr (Var)	72
estilo_inventario (Var)	72		
etapa_inventario (Var)	68, 68	I	
Examinar (Acc)	37, 52, 52, 139	Identicos (Rut)	30, 91
ExaminarSub (Rut)	37	IdiomaAInformes (Rut)	91
Excavar (Acc)	43	IdiomaContraccion (Rut)	92
EXCEPTO_PE (Cons)	101	IdiomaContracciones (Cons)	109
		IdiomaDirecciones (Rut)	92
F		IdiomaEsVerbo (Rut)	92
Facilitar.h (Fich)	10	IdiomaGeneroAnimado (Cons)	109
false (Cons)	11, 13, 19–24, 29, 31, 50, 51, 54, 58, 60, 64, 68, 78, 87, 88, 98, 98, 101	IdiomaGeneroInanimado (Cons)	109
femenina (Atr)	13	IdiomaHoraDelDia (Rut)	92
femenino (Atr)	13, 13, 14, 23, 24, 27, 28, 79, 80	IdiomaImprimirComando (Rut)	92
Fijar (Acc)	43	IdiomaImprimirNombreCorto (Rut)	92
FijarPronombre (Rut)	84, 84	IdiomaNumero (Rut)	92
FiltrarMulti (Rut)	91	IdiomaPreguntarPrep (Rut)	92
filtro_token (Var)	72	IdiomaVerbo (Rut)	88, 92
FiltroUsuario (Rut)	91	imperativo (Prop)	25, 95
Finalizar (Acc)	39	ImpNumAlin (Rut)	92
Frotar (Acc)	10, 43	Imprimir_Espacios (Rut)	92
		ImprimirComando (Rut)	92
G		ImprimirIrregular (Rut)	92
G_FEMENINO (Cons)	23, 24, 98	ImprimirOEjacular (Rut)	84
G_MASCULINO (Cons)	23, 24, 98	ImprimirRango (Rut)	88, 109
G_PLURAL (Cons)	23, 24, 98	ImprimirTareas (Rut)	88, 96
general (Atr)	13, 27	ImprimirVerbo (Rut)	88
genero (Prop)	13, 14, 23, 24, 27, 28, 79–81, 98	indef_casos (Var)	72
Gesticular (Acc)	43	indef_esperado (Var)	72
Girar (Acc)	43	indef_numero_en (Var)	72
gramatica (Prop)	24	indef_posibambig (Var)	72
Gramatica.h (Fich)	5, 9, 10, 37	indef_propietario (Var)	72
gramatica_normal_tras (Var)	72	indef_suponer_p (Var)	72
Grammar__Version (Cons)	109	indef_tipo (Var)	72
		InDefArt (Rut)	84
H		IndefArt (Rut)	84
hablable (Atr)	13, 24, 28	indentacion_eld (Var)	72
		INDENTAR_BIT (Cons)	99

indirect (Rut)	81	localizacion_actor (Var)	73
INFIX (Cons)	96	localizacion_real (Var)	57, 68, 69, 77
infixe.h (Fich)	10	long_encajado (Var)	73
INFOPARCIAL_BIT (Cons)	99	LongitudDePalabra (Rut)	85
INFOTOTAL_BIT (Cons)	100	LoSiento (Acc)	45
inicial (Prop)	25, 26, 57, 58, 85	LowKey_Menu (Rut)	92
Inicializar (Rut)	40, 68, 69, 88	Lugares (Acc)	16, 40, 96
IniciarPregunta (Rut)	92	Lugares1Sub (Rut)	92
InsertarLetraBuffer (Rut)	92	LugarNuevo (Rut)	57, 89
IntentarCogerObjeto (Rut)	92	luz (Atr)	10, 13, 15
IntentarEncajarObjeto (Rut)	92		
IntentarNumero (Rut)	84	M	
InterpretarNombre (Rut)	29, 30, 88, 109	M__L (Rut)	92
InterpretarNumero (Rut)	88	M_L (Rut)	92
InterpretarToken (Rut)	92	Main (Rut)	77, 92
interprete_estandar (Var)	72	mant_np (Var)	73
Inv (Acc)	53	MasAlla (Rut)	89
InvAlto (Acc)	53	masculino (Atr)	13, 13
InvAncho (Acc)	53	MAX_RELOJES (Cons)	96
Inventario (Acc)	70	MejorIntuicion (Rut)	92
Ir (Acc)	52, 53, 53, 56, 61	MensajeMuerte (Rut)	67, 89
IrAmbiguo (Acc)	44	Mensajes.h (Fich)	5, 10
IrDonde (Acc)	63	MensajesLibreria (Obj)	64, 65, 77, 88, 89, 100, 101
irrelevante (Prop)	26	meta (Var)	73
itobj (Var)	72	metaclass (Rut)	81
		Meter (Acc)	37, 38, 55, 55, 56, 58, 62, 65
J		Meterse (Acc)	56, 56
JUGADAS__TX (Cons)	104	MI_BIT (Cons)	109
jugador (Var)	67, 68, 78, 89	Mirar (Acc)	37, 56, 84, 89
JugadorA (Rut)	84	MirarDebajo (Acc)	45
jugar (Prop)	77	MirarHaciaSub (Rut)	89
just_undone (Var)	73	Miscelanea (Acc)	64, 100, 102
		ml_n (Var)	73, 77, 100
K		ml_o (Var)	73, 77
KKFINAL_PE (Cons)	101	MLIdioma (Rut)	64, 92
		MMULTI_PE (Cons)	101
L		modo_indef (Var)	73
la_hora (Var)	68, 85	modo_mantenido (Var)	73
Lanzar (Acc)	35, 44, 44, 55, 65	modo_multi (Var)	73
laoscuridad (Obj)	57, 68, 77, 105	modo_noposeido (Var)	73
lastdesc (Var)	73	modo_notificar (Var)	69
Leer (Acc)	52	modo_transcripcion (Var)	73
LeerComandos (Acc)	63	ModoM1 (Acc)	40
LibreriaInform (Obj)	77	ModoM2 (Acc)	40
lineaEstado1 (Var)	73	ModoM3 (Acc)	40
lineaEstado2 (Var)	73	modomirar (Var)	16, 21, 40, 58, 73
ListaMiscelanea (Acc)	64, 64	Mostrar (Acc)	35, 58
listando_junto (Var)	73	MostrarObjeto (Acc)	63
listar_juntos (Prop)	26, 68	MostrarVerbo (Acc)	63
ListarIguales (Rut)	92	MoverJugador (Rut)	92
listarse (Prop)	26, 27, 68	MoverObjetosFlotantes (Rut)	92
Llenar (Acc)	44	MoverPalabra (Rut)	93
LLEVAR_MAX (Cons)	14, 20, 96, 97	movido (Atr)	13
Local (Rut)	58, 84	Msg1Ph (Fich)	10
localizacion (Var)	22, 54, 57, 68, 77, 88, 89	multi_contexto (Var)	73

multi_esperado (Var) 73
 multi_hallado (Var) 73
 MULTI_PE (Cons) 102
 MUYPOCO_PE (Cons) 102

N

NADA_TX (Cons) 104
 NADA_PE (Cons) 102
 Nadar (Acc) 45
 name (Prop) 27
 neutro (Atr) 14
 nextbest_tipoerror (Var) 73
 NivelTraza (Acc) 63
 No (Acc) 45, 45, 47, 70
 NO1_WD (Cons) 45, 104
 NO2_WD (Cons) 45, 104
 NO3_WD (Cons) 45, 105
 no_deducir (Var) 73
 no_interpretar (Var) 69
 NO_LUGARES (Cons) 13, 16, 40, 96
 no_poseido (Var) 73
 no_puedes_ir (Prop) 28, 54
 NO_PUNTUACION (Cons) 39, 40, 96
 NoComprendido (Acc) 25, 64, 64, 101
 NoLoVeo (Rut) 93
 nombre (Prop) 14, 18, 25, 26, 27, 28, 30, 31, 64, 82, 85, 91
 nombre_corto (Prop) 24, 27, 28, 80
 nombre_corto_indef (Prop) 28
 nombre_elemento (Var) 73
 nombre_f (Prop) 24, 27, 28, 81, 88, 91
 nombre_fp (Prop) 24, 27, 28, 81, 88, 91
 nombre_m (Prop) 24, 27, 28, 81, 82, 88
 nombre_mp (Prop) 24, 27, 28, 81, 88, 91
 nombrequelplural (Atr) 14, 23, 24, 28, 79–81
 nombrequesado (Atr) 14
 NOPUEDESIR_TX (Cons) 105
 NoSoportaAlJugador (Rut) 93
 nothing (Cons) 98
 NOTIENES_PE (Cons) 102
 NotificarLaPuntuacion (Rut) 93
 NOVEO_PE (Cons) 102
 np (Var) 46, 47, 69, 83, 86
 nsns (Var) 73
 NUEVALINEA_BIT (Cons) 100
 NULL (Cons) 87, 98
 Num_DirDicc (Rut) 93
 num_palabras (Var) 73
 numero_de_clases (Var) 73
 numero_de_encajados (Var) 74
 numero_especial (Var) 74
 numero_especial1 (Var) 74
 numero_especial2 (Var) 74
 numero_interpretado (Var) 46, 74
 NUMERO_PE (Cons) 102

O

O_TX (Cons) 105
 obj_abajo (Obj) 58, 62, 77, 78
 obj_adentro (Obj) 77
 obj_afuera (Obj) 61, 77, 78
 obj_arriba (Obj) 77, 78
 obj_dentro (Obj) 78
 obj_e (Obj) 77, 78
 obj_n (Obj) 32, 53, 54, 77, 78, 89
 obj_ne (Obj) 77, 78
 obj_no (Obj) 77, 78
 obj_o (Obj) 77, 78
 obj_s (Obj) 32, 53, 77, 78, 89
 obj_se (Obj) 77, 78
 obj_so (Obj) 77, 78
 objeto_pronombre (Var) 74
 objeto_raiz (Var) 74
 OBJETO_SACO (Cons) 50, 97
 ObjetoEnAlcancePorAlgo (Rut) 93
 ObjetoEsIntocable (Rut) 85, 93
 Objetos (Acc) 13, 40, 96
 Objetos1Sub (Rut) 93
 objjugador (Obj) 20, 67, 68, 78
 ObtenerGenero (Rut) 93
 ObtenerGNADeObjeto (Rut) 93
 oculta (Atr) 14
 OCULTAR_BIT (Cons) 100
 oculto (Atr) 14, 54, 57
 old_herobj (Var) 74
 old_himobj (Var) 74
 old_itobj (Var) 74
 Oler (Acc) 9, 19, 45
 OlerSub (Rut) 9
 OOPS1_WD (Cons) 105
 OOPS2_WD (Cons) 105
 OOPS3_WD (Cons) 105
 Orden (Acc) 29, 35, 64, 64
 OrdenarJuntos (Rut) 93
 OrdenarLista (Rut) 93
 ordenes (Prop) 25, 28, 28, 29, 32, 35, 46, 64, 101
 OSCURIDAD_TX (Cons) 57, 105
 OTRA VEZ1_WD (Cons) 105
 OTRA VEZ2_WD (Cons) 105
 OTRA VEZ3_WD (Cons) 105
 otro (Var) . 19, 24, 25, 29, 31, 38, 42, 44–47, 62, 64, 69, 69, 86
 OTRO1_WD (Cons) 105
 OTRO2_WD (Cons) 105
 OTRO3_WD (Cons) 105
 OTRO_BIT (Cons) 109

P

paa_tfl (Var)	74	PruebaDeAlcance (Rut)	86
palabra_especial (Var)	74	PSN__ (Rut)	93
palabra_pronombre (Var)	74	puerta (Atr)	11, 15, 22, 31
palabra_verbo (Var)	24, 25, 74	puerta_a (Prop)	15, 22, 31, 54
palabra_verbonum (Var)	24, 74	puesta (Atr)	15
PalabraEnPropiedad (Rut)	85	puesto (Atr)	14, 15, 15, 51, 59, 62
PalabraNumero (Rut)	93	punt_anterior (Var)	75
PalabraSustantivo (Rut)	93	puntos_cosas (Var)	75
parametros (Var)	74	PUNTOS_LUGAR (Cons)	15, 58, 97
parametros_deseados (Var)	74	PUNTOS_OBJETO (Cons)	15, 97
PararDaemon (Rut)	20, 85	puntos_sitios (Var)	75
PararReloj (Rut)	35, 85	Puntuacion (Acc)	40
parent (Rut)	81	puntuacion (Var)	70
parse (Var)	74	PUNTUACION1__WD (Cons)	106
parse2 (Var)	74	PUNTUACION2__WD (Cons)	106
parse_nombre (Prop)	29, 29, 30, 64, 69	PUNTUACION__TX (Cons)	106
parseaux (Var)	74	PUNTUACION_MAX (Cons)	97
Parser__parse (Rut)	78, 93	puntuacion_tareas (Var)	70, 88, 96
parser_dos (Var)	64, 74	PuntuacionListaEncajes (Rut)	93
parser_inflexion (Var)	74	PuntuacionLlegada (Rut)	58, 93
parser_listo (Var)	69	PuntuacionSub (Rut)	96
parser_trace (Var)	74	PuntuacionTotal (Acc)	40
parser_uno (Var)	64, 74		
ParserInform (Obj)	78	Q	
PasaElTiempo (Rut)	89	QKEY1__KY (Cons)	106
PATRON_NULO (Cons)	109	QKEY1__TX (Cons)	106
Pedir (Acc)	45	QKEY2__KY (Cons)	106
Pensar (Acc)	46	QKEY2__TX (Cons)	106
permitir_plurales (Var)	74	Quemar (Acc)	46, 52
PermitirEmpujarDir (Rut)	43, 85	QUIEN__TX (Cons)	106
plural (Prop)	26, 30, 30	quitacentos (Var)	75
PLURAL_BIT (Cons)	109	QuitandoRFinal (Rut)	93
PonerA (Acc)	43, 46	QuitarAcentos (Rut)	93
PonerAlAlcance (Rut)	34, 85	QuitarCerrojo (Acc)	12, 59, 70
PonerArticuloDelante (Rut)	93		
PonerLaHora (Rut)	85	R	
PonerSobre (Acc)	37, 55, 58, 58, 59, 62, 65	R_Process (Rut)	93
Predecible (Acc)	63	random (Rut)	81
PREGUNTAAMBITO_PE (Cons)	102	razon_alcance (Var)	75
PreguntaCualExactamente (Var)	74	RAZON_BUCLEALCANCE (Cons)	108
Preguntar (Acc)	35, 42, 44, 46	RAZON_CADA_TURN0 (Cons)	108
PreguntarPreposicion (Rut)	89	RAZON_HABLAR (Cons)	108
PreguntaSiNo (Var)	45, 47, 70, 70	RAZON_PARSING (Cons)	108
prenda (Atr)	14, 15, 27, 62	RAZON_REACCIONAR_ANTES (Cons)	108
prepdeducida (Var)	74	RAZON_REACCIONAR_DESPUES (Cons)	108
PREVIOYO__TX (Cons)	105	RAZON_TESTALCANCE (Cons)	108
Print__PName (Rut)	81	reaccionar_antes (Prop)	31, 31, 32, 42
Print_ScL (Rut)	93	reaccionar_despues (Prop)	32
PrintShortName (Rut)	86	Recibir (Acc)	38, 55, 56, 59, 65, 65
Probar (Acc)	46	RecibirLanzamiento (Acc)	44, 65, 65
Prompt (Acc)	65, 65	recipiente (Atr)	11, 13, 15, 20, 37, 38, 49, 50, 55, 56, 58, 61, 62
PRONOM_PE (Cons)	102	RECUPERAR__WD (Cons)	106
Pronombres (Acc)	40	RECURSIVO_BIT (Cons)	100
propio (Atr)	14, 79	regla_coger_todo (Var)	75

Reiniciar (Acc).....	40	si_encendida (Prop)	34
REINICIAR__WD (Cons)	106	si_encendido (Prop)	21, 33, 34, 34, 85, 100
ReiniciarPalabrasVagas (Rut)	93	sibling (Rut)	81
Release (Cons)	97	SIEMPRE_BIT (Cons)	100
relojes_activos (Var)	75	SiguienteElemento (Rut)	94
REPARSE_CODE (Cons)	83	SiguientePalabra (Rut) 29, 30, 46, 47, 69, 83, 86, 86	
Responder (Acc)	29, 35, 46, 46	SiguientePalabraParar (Rut)	86
Restaurar (Acc)	41	SIN_DIRECCIONES (Cons)	77, 97
Retorcer (Acc)	47	SINARTICULO_BIT (Cons)	100
RevisarMulti (Rut)	93	SiOno (Rut)	45, 70, 86
RevisionLib (Cons)	109	Soplar (Acc)	47
Rezar (Acc)	47	soporte (Atr) .. 15, 20, 37, 49, 50, 55, 56, 59, 61, 62	
RKEY__TX (Cons)	106	Soso (Acc)	47
RPG_CRIATURA (Cons)	109	STRICT_MODE (Cons)	97
RPG_FALLO (Cons)	109	Subir (Acc)	61
RPG_MULTI (Cons)	110	suma_al_alcance (Prop)	34, 34, 85
RPG_MULTIDENTRO (Cons)	110	SumarAlAlcance (Rut)	86, 87
RPG_MULTIEXCEPTO (Cons)	110	SustraerMulti (Rut)	94
RPG_MULTIPOSEIDO (Cons)	110		
RPG_NOMBRE (Cons)	110	T	
RPG_NUMERO (Cons)	110	Tacos (Acc)	39, 47
RPG_POSEIDO (Cons)	110	TAMANIO_LISTA_ENCAJAN (Cons)	110
RPG_PREPOSICION (Cons)	110	tamano_listando (Var)	75
RPG_REPARSE (Cons)	110	tate_callao (Var)	70, 70
RutinaMirar (Rut)	58, 89	tdatos_encontrado (Var)	75
RutinaPostJuego (Rut)	89	techo_de_visibilidad (Var)	75
RutinaPreJuego (Rut)	89	TECLAANT1__KY (Cons)	107
RutinasAntes (Rut)	93	TECLAANT2__KY (Cons)	107
RutinasDespues (Rut)	86, 94	TECLAANT__TX (Cons)	107
		Teclado (Rut)	86
S		TECLASIG1__KY (Cons)	107
Sacar (Acc)	60, 60	TECLASIG2__KY (Cons)	107
Salidas (Acc)	1, 5, 32, 60, 139	TECLASIG__TX (Cons)	107
salidas (Prop)	32, 32, 60	TERMINAR1__WD (Cons)	107
Salir (Acc)	53, 56, 61, 61	TERMINAR2__WD (Cons)	107
Salirse (Acc)	61, 61	TestCriatura (Rut)	94
Saltar (Acc)	47	tiempo_agotado (Prop)	34, 35, 82, 96
SaltarSobre (Acc)	47, 47	TIEMPO_REAL (Cons)	69
Salvar (Acc)	41	tiempo_restante (Prop)	35, 82, 85
SALVO1__WD (Cons)	106	TieneFuenteDeLuz (Rut)	87
SALVO2__WD (Cons)	106	tipoerror (Var)	25, 29, 64, 75, 101
SALVO3__WD (Cons)	106	Tirar (Acc)	43, 44, 48
SeRefiere (Rut)	94	Titular (Cons)	97, 98
Serial (Cons)	97	Tocar (Acc)	48
SeVeATraves (Rut)	94	TODO1__WD (Cons)	107
Si (Acc)	45, 47	TODO2__WD (Cons)	107
SI1__WD (Cons)	45, 106	TODO3__WD (Cons)	107
SI2__WD (Cons)	45, 107	TODO4__WD (Cons)	107
SI3__WD (Cons)	45, 107	TODO5__WD (Cons)	107
si_abierta (Prop)	32	token_alcance (Var)	75
si_abierto (Prop)	21, 32, 32, 34, 58, 85, 100	TOKEN_CRIATURA (Cons)	110
si_apagada (Prop)	33	TOKEN_ESPECIAL (Cons)	110
si_apagado (Prop)	21, 33, 33, 34, 85, 100	TOKEN_FINAL (Cons)	110
si_cerrada (Prop)	33	TOKEN_MULTI (Cons)	110
si_cerrado (Prop)	21, 33, 33, 58, 85, 100	TOKEN_MULTIDENTRO (Cons)	110

TOKEN_MULTIEXCEPTO (Cons)	110	ValorDelPronombre (Rut)	87
TOKEN_MULTIPOSEIDO (Cons)	110	ValorOEjecutar (Rut)	87
TOKEN_NOMBRE (Cons)	110	VERBO_PE (Cons)	102
TOKEN_NUMERO (Cons)	110	VerboDesconocido (Rut)	90
TOKEN_POSEIDO (Cons)	110	Verificar (Acc)	41
TOKEN_TEMA (Cons)	111	VersionIdioma (Cons)	111
Tokenise__ (Rut)	94	Vestir (Acc)	62
TopeAlcanzable (Rut)	94	vida (Prop) 11, 29, 32, 35, 35, 42, 44, 46, 47, 51, 58,	
Transferir (Acc)	61	64, 67	
transparente (Atr)	13, 15	visitado (Atr)	16
TrasElPrompt (Rut)	89		
TrazarAccion (Rut)	94	W	
Trepar (Acc)	48, 56	workflag (Atr)	11
true (Cons) ... 11, 13, 19–24, 29, 35, 38, 44, 46, 48,		X	
50–52, 54, 55, 58, 59, 63, 65, 68, 69, 78,		x_cuenta_ambito (Var)	75
84–87, 98, 98, 111		XArbol (Acc)	63
TT_ALCANCE (Cons)	111	xcommsdir (Var)	75
TT_ELEMENTAL (Cons)	111	XCompruebaMover (Rut)	94
TT_FILTRO_ATRIB (Cons)	111	XIrA (Acc)	63
TT_FILTRO Rutina (Cons)	111	XMover (Acc)	63
TT_ILEGAL (Cons)	111	XObj (Rut)	94
TT_PREPOSICION (Cons)	111	XRobar (Acc)	63
TT_RPG (Cons)	111	Y	
ttipo_encontrado (Var)	75	Y1__WD (Cons)	107
TUMISMO__TX (Cons)	107	Y2__WD (Cons)	107
turnos (Var)	70	Y3__WD (Cons)	107
U		Y__TX (Cons)	108
UnaDireccion (Rut)	94	YANOPRON_PE (Cons)	102
uno (Var) . 24, 25, 29, 31, 38, 44–47, 51, 53, 54, 58,		YO1__WD (Cons)	108
59, 64, 70, 70		YO2__WD (Cons)	108
V		YO3__WD (Cons)	108
Vaciar (Acc)	62	Z	
VaciarEn (Acc)	62, 62	ZRegion (Rut)	87
valepuntos (Atr)	15, 58, 97		
valor_lj (Var)	75		

10.2. Equivalencias Inglés - Español

Los dos siguientes índices listan todos los identificadores tanto en inglés como en español, y por tanto pueden usarse a modo de diccionario. Además se indica la página en la cual el correspondiente identificador es descrito en este documento. Los identificadores originales en inglés aparecen en *texto inclinado*, y los identificadores en español en texto normal.

Bajo la entrada “Nuevos identificadores” (de la sección Español-Inglés) se listan los identificadores que sólo existen en la versión española de la librería y por tanto no tienen original en inglés.

Símbolos

<i>(The)</i> (Rut): (<i>_El</i>)	79
<i>(a)</i> (Rut): (<i>un</i>)	80
<i>(name)</i> (Rut): (<i>_nombre_</i>)	80
<i>(number)</i> (Rut): (<i>numero</i>)	80
<i>(the)</i> (Rut): (<i>el</i>)	79

A

<i>absent</i> (Atr): ausente	11
<i>Achieved</i> (Rut): Conseguido	82
<i>action</i> (Var): accion	67
<i>action_reversed</i> (Var): accion_invertida	70
<i>action_to_be</i> (Var): accion_que_seria	71
<i>ActionsOff</i> (Acc): DesactivarAcciones	63
<i>ActionsOn</i> (Acc): ActivarAcciones	62
<i>active_timers</i> (Var): relojes_activos	75
<i>actor</i> (Var): actor	67
<i>actors_location</i> (Var): localizacion_actor	73
<i>add_to_scope</i> (Prop): suma_al_alcance	34
<i>AddToScope</i> (Rut): SumarAlAlcance	86
<i>ADirection</i> (Rut): UnaDireccion	94
<i>AdjustLight</i> (Rut): AjustarLuz	90
<i>advance_warning</i> (Var): aviso_avanzar	71
<i>after</i> (Prop): despues	21
<i>AfterLife</i> (Rut): MasAlla	89
<i>AfterPrompt</i> (Rut): TrasElPrompt	89
<i>AfterRoutines</i> (Rut): RutinasDespues	86, 94
<i>AGAIN1__WD</i> (Cons): OTRA VEZ1__WD	105
<i>AGAIN2__WD</i> (Cons): OTRA VEZ2__WD	105
<i>AGAIN3__WD</i> (Cons): OTRA VEZ3__WD	105
<i>ALL1__WD</i> (Cons): TODO1__WD	107
<i>ALL2__WD</i> (Cons): TODO2__WD	107
<i>ALL3__WD</i> (Cons): TODO3__WD	107
<i>ALL4__WD</i> (Cons): TODO4__WD	107
<i>ALL5__WD</i> (Cons): TODO5__WD	107
<i>allow_plurals</i> (Var): permitir_plurales	74
<i>AllowPushDir</i> (Rut): PermitirEmpujarDir	85
<i>Amusing</i> (Rut): Curiosidades	87
<i>AMUSING__WD</i> (Cons): CURIOSIDADES__WD	103

AMUSING_PROVIDED (Cons): HAY_CURIOSIDADES

	95
<i>AnalyseToken</i> (Rut): AnalizarToken	90
<i>AND1__WD</i> (Cons): Y1__WD	107
<i>AND2__WD</i> (Cons): Y2__WD	107
<i>AND3__WD</i> (Cons): Y3__WD	107
<i>AND__TX</i> (Cons): Y__TX	108
<i>ANIMA_PE</i> (Cons): ANIMA_PE	101
<i>animate</i> (Atr): animado	11
<i>Answer</i> (Acc): Responder	46
<i>ARE2__TX</i> (Cons): HAYP2__TX	104
<i>ARE__TX</i> (Cons): HAYP__TX	104
<i>article</i> (Prop): articulo	19
<i>articles</i> (Prop): articulos	19
<i>Ask</i> (Acc): Preguntar	46
<i>AskFor</i> (Acc): Pedir	45
<i>ASKSCOPE_PE</i> (Cons): PREGUNTAAM-BITO_PE	102
<i>ats_flag</i> (Var): bandera_paa	71
<i>ats_hls</i> (Var): paa_tfl	74
<i>Attack</i> (Acc): Atacar	41
<i>AttemptToTakeObject</i> (Rut): IntentarCogerObjeto	92
<i>ATTR_FILTER_TT</i> (Cons): TT_FILTRO_ATTRIB	111

B

<i>Banner</i> (Rut): Anuncio	90
<i>before</i> (Prop): antes	18
<i>BeforeParsing</i> (Rut): AntesDelParsing	87
<i>BeforeRoutines</i> (Rut): RutinasAntes	93
<i>best_etype</i> (Var): best_tipoerror	71
<i>BestGuess</i> (Rut): MejorIntuicion	92
<i>Blow</i> (Acc): Soplar	47
<i>buffer</i> (Var): buffer	71
<i>buffer2</i> (Var): buffer2	71
<i>buffer3</i> (Var): buffer3	71
<i>Burn</i> (Acc): Quemar	46
<i>BUT1__WD</i> (Cons): SALVO1__WD	106
<i>BUT2__WD</i> (Cons): SALVO2__WD	106
<i>BUT2__WD</i> (Cons): SALVO3__WD	106

Buy (Acc): Comprar	42	description (Prop): descripcion	21
C		Descriptors (Rut): Descriptores	91
c_style (Var): estilo_ac	72	DictionaryLookup (Rut): BuscarEnDiccionario	82
cant_go (Prop): no_puedes_ir	28	Dig (Acc): Excavar	43
CANTGO__TX (Cons): NOPUEDESIR__TX	105	DisplayStatus (Rut): ActualizarEstatus	90
CantSee (Rut): NoLoVeo	93	Disrobe (Acc): Desvestir	51
CANTSEE_PE (Cons): NOVEO_PE	102	DoMenu (Rut): DoMenu	83
capacity (Prop): capacidad	20	dont_infer (Var): no_deducir	73
CDefArt (Rut): CDefArt	82	door (Atr): puerta	15
ChangeDefault (Rut): CambiarDefecto	82	door_dir (Prop): direcc_puerta	22
ChangePlayer (Rut): CambiarJugador	82	door_to (Prop): puerta_a	31
child (Rut): child	80	DoScopeAction (Rut): EfectuarAccionesAlcance	91
children (Rut): children	81	DrawStatusLine (Rut): DibujarLineaEstado	83
ChooseObjects (Rut): ElegirObjetos	88	Drink (Acc): Beber	41
Climb (Acc): Tregar	48	Drop (Acc): Dejar	51
Close (Acc): Cerrar	49	Dword_No (Rut): DirDicc_Num	91
clothing (Atr): prenda	14	E	
CommandsOff (Acc): DesactivarComandos	63	e_obj (Obj): obj_e	78
CommandsOn (Acc): ActivarComandos	62	e_to (Prop): al_e	18
CommandsRead (Acc): LeerComandos	63	each_turn (Prop): cada_turno	20
CommonAncestor (Rut): AntepasadoComun	90	EACH_TURN_REASON (Cons): RAZON_CADA_TURN	108
compass (Obj): brujula	77	Eat (Acc): Comer	50
CONCEAL_BIT (Cons): OCULTAR_BIT	100	edible (Atr): comestible	12
concealed (Atr): oculto	14	ELEMENTARY__TT (Cons): TT_ELEMENTAL	111
Consult (Acc): Consultar	42	Empty (Acc): Vaciar	62
consult_from (Var): consultar_desde	71	EmptyT (Acc): VaciarEn	62
consult_words (Var): consultar_num_palabras	71	ENDIT_TOKEN (Cons): TOKEN_FINAL	110
container (Atr): recipiente	15	ENGLISH_BIT (Cons): ESPANOL_BIT	99
CopyBuffer (Rut): CopiarBuffer	90	EnglishNumber (Rut): EnglishNumber	83, 91
CREATURE_TOKEN (Cons): TOKEN_CRIATURA	110	Enter (Acc): Entrar	52
CreatureTest (Rut): TestCriatura	94	enterable (Atr): entrable	12
Cut (Acc): Cortar	42	etype (Var): tipoerror	75
D		Examine (Acc): Examinar	52
d_obj (Obj): obj_abajo	78	EXCEPT_PE (Cons): EXCEPTO_PE	101
d_to (Prop): abajo	17	Exit (Acc): Salir	61
daemon (Prop): daemon	20	F	
DARKNESS__TX (Cons): OSCURIDAD__TX	105	false (Cons): false	98
DarkToDark (Rut): CaminarAOscuras	87	female (Atr): femenino	13
deadflag (Var): banderafin	67	Fill (Acc): Llenar	44
DeathMessage (Rut): MensajeMuerte	89	FORMER__TX (Cons): PREVIOYO__TX	105
DEBUG (Cons): DEBUG	95	found_in (Prop): esta_en	22
debug_flag (Var): bandera_debug	71	found_tdata (Var): tdatos_encontrado	75
DebugAction (Rut): DepurarAccion	90	found_ttype (Var): ttipo_encontrado	75
DebugAttribute (Rut): DepurarAtributo	91	FULLINV_BIT (Cons): INFOTOTAL_BIT	100
DebugGrammarLine (Rut): DepurarLineaGramatica	91	FullScore (Acc): PuntuacionTotal	40
DebugParameter (Rut): DepurarParametro	91	FULLSCORE1__WD (Cons): PUNTUA-	106
DebugToken (Rut): DepurarToken	91	CION1__WD	
DefArt (Rut): DefArt	83	FULLSCORE__WD (Cons): PUNTUA-	106
DEFART_BIT (Cons): DEFINIDO_BIT	99	CION2__WD	
DEFART_PK (Cons): DEFINIDO_PK	109		
describe (Prop): describir	20		

G

<i>GamePostRoutine</i> (Rut): RutinaPostJuego	89	<i>indirect</i> (Rut): indirect	81
<i>GamePreRoutine</i> (Rut): RutinaPreJuego	89	<i>IndirectlyContains</i> (Rut): ContieneIndirectamente	90
<i>general</i> (Atr): general	13	<i>inferfrom</i> (Var): deducedesde	72
<i>GetGender</i> (Rut): ObtenerGenero	93	<i>inferword</i> (Var): prepdeducida	74
<i>GetGNAOfObject</i> (Rut): ObtenerGNADeObjeto	93	<i>INFIX</i> (Cons): INFIX	96
<i>GetOff</i> (Acc): Salirse	61	<i>infix.h</i> (Fich): infixe.h	10
<i>Give</i> (Acc): Dar	50	<i>InformLibrary</i> (Obj): LibreriaInform	77
<i>Go</i> (Acc): Ir	53	<i>InformParser</i> (Obj): ParserInform	78
<i>GoIn</i> (Acc): Meterse	56	<i>initial</i> (Prop): inicial	25
<i>Gonear</i> (Acc): IrDonde	63	<i>Initialise</i> (Rut): Inicializar	88
<i>Goto</i> (Acc): XIrA	63	<i>inp1</i> (Var): dat1	71
<i>GPR_CREATURE</i> (Cons): RPG_CRIATURA	109	<i>inp2</i> (Var): dat2	72
<i>GPR_FAIL</i> (Cons): RPG_FALLO	109	<i>InScope</i> (Rut): AlAlcance	87
<i>GPR_HELD</i> (Cons): RPG_POSEIDO	110	<i>Insert</i> (Acc): Meter	55
<i>GPR_MULTI</i> (Cons): RPG_MULTI	110	<i>inside_description</i> (Prop): descripcion_dentro	21
<i>GPR_MULTIEXCEPT</i> (Cons): RPG_MULTIEXCEPT	110	<i>Inv</i> (Acc): Inv	53
<i>GPR_MULTIHOLD</i> (Cons): RPG_MULTIOSEIDO	110	<i>invent</i> (Prop): listarse	26
<i>GPR_MULTIINSIDE</i> (Cons): RPG_MULTIDENTRO	110	<i>inventory_stage</i> (Var): etapa_inventario	68
<i>GPR_NOUN</i> (Cons): RPG_NOMBRE	110	<i>inventory_style</i> (Var): estilo_inventario	72
<i>GPR_NUMBER</i> (Cons): RPG_NUMERO	110	<i>InvTall</i> (Acc): InvAlto	53
<i>GPR_PREPOSITION</i> (Cons): RPG_PREPOSICION	110	<i>InvWide</i> (Acc): InvAncho	53
<i>GPR_REPARSE</i> (Cons): RPG_REPARSE	110	<i>IS2__TX</i> (Cons): HAY2__TX	104
<i>GPR_TT</i> (Cons): TT_RPG	111	<i>IS__TX</i> (Cons): HAY__TX	104
<i>grammar</i> (Prop): gramatica	24	<i>ISARE_BIT</i> (Cons): HAY_BIT	99
<i>Grammar__Version</i> (Cons): Grammar__Version	109	<i>IsSeeThrough</i> (Rut): SeVeATraves	94

H

</

I

<i>Identical</i> (Rut): Identicos	91	<i>Kiss</i> (Acc): Besar	41
<i>ILLEGAL_TT</i> (Cons): TT_ILEGAL.....	111	L	
<i>in_obj</i> (Obj): obj_dentro	78	<i>L__M</i> (Rut): M__L	92
<i>in_to</i> (Prop): adentro	17	<i>L_M</i> (Rut): M_L	92
<i>indef_cases</i> (Var): indef_casos	72	<i>LanguageAnimateGender</i> (Cons): IdiomaGen- eroAnimado	109
<i>indef_guess_p</i> (Var): indef_suponer_p.....	72	<i>LanguageCases</i> (Cons): CasosLenguaje	109
<i>indef_mode</i> (Var): modo_indef	73	<i>LanguageContraction</i> (Rut): IdiomaContraccion	92
<i>indef_nspec_at</i> (Var): indef_numero_en	72	<i>LanguageContractionForms</i> (Cons): IdiomaCon- tracciones	109
<i>indef_owner</i> (Var): indef_propietario	72	<i>LanguageDirection</i> (Rut): IdiomaDirecciones	92
<i>indef_possambig</i> (Var): indef_posibambig	72	<i>LanguageInanimateGender</i> (Cons): IdiomaGen- eroInanimado	109
<i>indef_type</i> (Var): indef_tipo	72		
<i>indef_wanted</i> (Var): indef_esperado.....	72		
<i>InDefArt</i> (Rut): InDefArt.....	84		
<i>INDENT_BIT</i> (Cons): INDENTAR BIT.....	99		

<i>LanguageIsVerb</i> (Rut): IdiomaEsVerbo.....	92	<i>male</i> (Atr): masculino	13
<i>LanguageLM</i> (Rut): MLIdioma.....	92	<i>match_from</i> (Var): encajado_desde	72
<i>LanguageNumber</i> (Rut): IdiomaNumero.....	92	<i>match_length</i> (Var): long_encajado	73
<i>LanguagePrintCommand</i> (Rut): IdiomaImprimir- Comando.....	92	<i>MATCH_LIST_SIZE</i> (Cons): TAMANIO_LISTA_ENCAJAN 110	
<i>LanguagePrintShortName</i> (Rut): IdiomaImprim- irNombreCorto	92	<i>MAX_CARRIED</i> (Cons): LLEVAR_MAX	96
<i>LanguageTimeOfDay</i> (Rut): IdiomaHoraDelDia	92	<i>MAX_SCORE</i> (Cons): PUNTUACION_MAX..	97
<i>LanguageToInformese</i> (Rut): IdiomaAInformes	91	<i>MAX_TIMERS</i> (Cons): MAX_RELOJES	96
<i>LanguageVerb</i> (Rut): IdiomaVerbo.....	92	<i>ME1__WD</i> (Cons): YO1__WD	108
<i>LanguageVersion</i> (Cons): VersionIdioma	111	<i>ME2__WD</i> (Cons): YO2__WD	108
<i>last_score</i> (Var): punt_anterior.....	75	<i>ME3__WD</i> (Cons): YO3__WD	108
<i>lastdesc</i> (Var): lastdesc.....	73	<i>menu_item</i> (Var): elemento_menu	72
<i>LetGo</i> (Acc): DejarSalir	63	<i>menu_nesting</i> (Var): anidacion_menu	71
<i>LibraryMessages</i> (Obj): MensajesLibreria	77	<i>meta</i> (Var): meta	73
<i>LibRelease</i> (Cons): RevisionLib	109	<i>metaclass</i> (Rut): metaclass	81
<i>LibSerial</i> (Cons): NumSerieLib	109	<i>Mild</i> (Acc): Soso.....	47
<i>life</i> (Prop): vida	35	<i>Miscellany</i> (Acc): Miscelanea	64
<i>light</i> (Atr): luz	13	<i>MMULTI_PE</i> (Cons): MMULTI_PE.....	101
<i>lightflag</i> (Var): banderaluz	71	<i>moved</i> (Atr): movido	13
<i>linklpa.h</i> (Fich): enlazlpa.h.....	10	<i>MoveFloatingObjects</i> (Rut): MoverObjetos- Flotantes	92
<i>linklv.h</i> (Fich): enlazlv.h.....	10	<i>MovePlayer</i> (Rut): MoverJugador.....	92
<i>list_together</i> (Prop): listar_juntos	26	<i>MOVES__TX</i> (Cons): JUGADAS__TX.....	104
<i>Listen</i> (Acc): Escuchar	43	<i>MoveWord</i> (Rut): MoverPalabra.....	93
<i>ListEqual</i> (Rut): ListarIguales	92	<i>multi_context</i> (Var): multi_contexto.....	73
<i>listing_size</i> (Var): tamaño_listando	75	<i>multi_had</i> (Var): multi_hallado	73
<i>listing_together</i> (Var): listando_junto	73	<i>multi_mode</i> (Var): modo_multi	73
<i>ListMiscellany</i> (Acc): ListaMiscelanea	64	<i>MULTI_PE</i> (Cons): MULTI_PE.....	102
<i>LIT_BIT</i> (Cons): ENCENDIDO_BIT.....	109	<i>MULTI_TOKEN</i> (Cons): TOKEN_MULTI....	110
<i>lm_n</i> (Var): ml_n.....	73	<i>multi_wanted</i> (Var): multi_esperado	73
<i>lm_o</i> (Var): ml_o.....	73	<i>MultiAdd</i> (Rut): AnadirMulti	90
<i>LMode1</i> (Acc): ModoM1	40	<i>MULTIEXCEPT_TOKEN</i> (Cons): TOKEN_MULTIEXCEPTO 110	
<i>LMode2</i> (Acc): ModoM2	40	<i>MultiFilter</i> (Rut): FiltrarMulti	91
<i>LMode3</i> (Acc): ModoM3	40	<i>multiflag</i> (Var): banderamulti.....	71
<i>Locale</i> (Rut): Local	84	<i>MULTIHELD_TOKEN</i> (Cons): TOKEN_MULTIPOSEIDO 110	
<i>location</i> (Var): localizacion.....	68	<i>MULTIINSIDE_TOKEN</i> (Cons): TOKEN_MULTIDENTRO 110	
<i>Lock</i> (Acc): EcharCerrojo	51	<i>MultiSub</i> (Rut): SustraerMulti	94
<i>lockable</i> (Atr): cerrojo	11	<i>MY_BIT</i> (Cons): MI_BIT	109
<i>locked</i> (Atr): cerrojoechado	12		
<i>Look</i> (Acc): Mirar	56	N	
<i>lookahead</i> (Var): elsiguiente	72	<i>n_obj</i> (Obj): obj_n	78
<i>lookmode</i> (Var): modomirar	73	<i>n_to</i> (Prop): al_n	18
<i>LookRoutine</i> (Rut): RutinaMirar	89	<i>name</i> (Prop): nombre	27
<i>LookUnder</i> (Acc): MirarDebajo	45	<i>ne_obj</i> (Obj): obj_ne	78
<i>LoopOverScope</i> (Rut): BucleAlcance	82	<i>ne_to</i> (Prop): al_ne	18
<i>LOOPOVERSCOPE_REASON</i> (Cons): RA- ZON_BUCLEALCANCE	108	<i>neuter</i> (Atr): neutro	14
<i>LowKey_Menu</i> (Rut): LowKey_Menu	92	<i>NEWLINE_BIT</i> (Cons): NUEVALINEA_BIT	100
<i>lt_value</i> (Var): valor_lj.....	75	<i>NewRoom</i> (Rut): LugarNuevo	89
M		<i>nextbest_etype</i> (Var): nextbest_tipoerror.....	73
<i>Main</i> (Rut): Main	92	<i>NextEntry</i> (Rut): SiguienteElemento	94
<i>Make__PN</i> (Cons): Crear__IN	109	<i>NextWord</i> (Rut): SiguientePalabra	86
<i>Make__PR</i> (Cons): Crear__IR	109	<i>NextWordStopped</i> (Rut): SiguientePalabraParar	86
<i>MakeMatch</i> (Rut): HacerEncaje	91		

<i>NKEY1__KY</i> (Cons): <i>TECLASIG1__KY</i>	107	<i>ObjectScopedBySomething</i> (Rut): <i>ObjetoEnAlcan-</i>	
<i>NKEY2__KY</i> (Cons): <i>TECLASIG2__KY</i>	107	<i>cePorAlgo</i>	93
<i>NKEY__TX</i> (Cons): <i>TECLASIG__TX</i>	107	<i>OF1__WD</i> (Cons): <i>DE1__WD</i>	103
<i>No</i> (Acc): <i>No</i>	45	<i>OF2__WD</i> (Cons): <i>DE2__WD</i>	103
<i>NO1__WD</i> (Cons): <i>NO1__WD</i>	104	<i>OF3__WD</i> (Cons): <i>DE3__WD</i>	103
<i>NO2__WD</i> (Cons): <i>NO2__WD</i>	104	<i>OF4__WD</i> (Cons): <i>DE4__WD</i>	103
<i>NO3__WD</i> (Cons): <i>NO3__WD</i>	105	<i>OffersLight</i> (Rut): <i>HayLuz</i>	84
<i>No__Dword</i> (Rut): <i>Num__DirDicc</i>	93	<i>old_herobj</i> (Var): <i>old_herobj</i>	74
<i>no_parse</i> (Var): <i>no_interpretar</i>	69	<i>old_himobj</i> (Var): <i>old_himobj</i>	74
<i>NO_PLACES</i> (Cons): <i>NO_LUGARES</i>	96	<i>old_itobj</i> (Var): <i>old_itobj</i>	74
<i>NOARTICLE_BIT</i> (Cons): <i>SINARTICULO_BIT</i>		<i>on</i> (Atr): <i>encendido</i>	12
100		<i>onotheld_mode</i> (Var): <i>amodo_noposeido</i>	71
<i>not_holding</i> (Var): <i>no_poseido</i>	73	<i>OOPS1__WD</i> (Cons): <i>OOPS1__WD</i>	105
<i>NoteArrival</i> (Rut): <i>AnotarLlegada</i>	90	<i>OOPS2__WD</i> (Cons): <i>OOPS2__WD</i>	105
<i>NoteObjectAcquisitions</i> (Rut): <i>AnotarObjeto-</i>		<i>OOPS3__WD</i> (Cons): <i>OOPS3__WD</i>	105
<i>Obtenidos</i>	90	<i>oops_from</i> (Var): <i>eepe_desde</i>	72
<i>notheld_mode</i> (Var): <i>modo_noposeido</i>	73	<i>Open</i> (Acc): <i>Abrir</i>	48
<i>NOTHELD_PE</i> (Cons): <i>NOTIENES_PE</i>	102	<i>open</i> (Atr): <i>abierto</i>	11
<i>nothing</i> (Cons): <i>nothing</i>	98	<i>openable</i> (Atr): <i>abrible</i>	11
<i>NOTHING__TX</i> (Cons): <i>NADA__TX</i>	104	<i>OR__TX</i> (Cons): <i>O__TX</i>	105
<i>NOTHING_PE</i> (Cons): <i>NADA_PE</i>	102	<i>Order</i> (Acc): <i>Orden</i>	64
<i>notify_mode</i> (Var): <i>modo_notificar</i>	69	<i>orders</i> (Prop): <i>ordenes</i>	28
<i>NotifyOff</i> (Acc): <i>DesactivarNotificacion</i>	39	<i>OTHER1__WD</i> (Cons): <i>OTRO1__WD</i>	105
<i>NotifyOn</i> (Acc): <i>ActivarNotificacion</i>	39	<i>OTHER2__WD</i> (Cons): <i>OTRO2__WD</i>	105
<i>NotifyTheScore</i> (Rut): <i>NotificarLaPuntuacion</i>	93	<i>OTHER3__WD</i> (Cons): <i>OTRO3__WD</i>	105
<i>NotSupportingThePlayer</i> (Rut): <i>NoSoportaAlJu-</i>		<i>OTHER_BIT</i> (Cons): <i>OTRO_BIT</i>	109
<i>gador</i>	93	<i>out_obj</i> (Obj): <i>obj_afuera</i>	78
<i>NotUnderstood</i> (Acc): <i>NoComprendido</i>	64	<i>out_to</i> (Prop): <i>afuera</i>	18
<i>noun</i> (Var): <i>uno</i>	70		
<i>NOUN_TOKEN</i> (Cons): <i>TOKEN_NOMBRE</i>	110	P	
<i>NounDomain</i> (Rut): <i>DominioNombre</i>	83	<i>PANum</i> (Rut): <i>ImpNumAlin</i>	92
<i>NounWord</i> (Rut): <i>PalabraSustantivo</i>	93	<i>parameters</i> (Var): <i>parametros</i>	74
<i>nsns</i> (Var): <i>nsns</i>	73	<i>params_wanted</i> (Var): <i>parametros_deseados</i>	74
<i>NULL</i> (Cons): <i>NULL</i>	98	<i>parent</i> (Rut): <i>parent</i>	81
<i>NULL_PATTERN</i> (Cons): <i>PATRON_NULO</i>	109	<i>parse</i> (Var): <i>parse</i>	74
<i>num_words</i> (Var): <i>num_palabras</i>	73	<i>parse2</i> (Var): <i>parse2</i>	74
<i>number</i> (Prop): <i>cantidad</i>	20	<i>parse_name</i> (Prop): <i>parse_nombre</i>	29
<i>number_matched</i> (Var): <i>numero_de_encajados</i>	74	<i>parsed_number</i> (Var): <i>numero_interpretado</i>	74
<i>number_of_classes</i> (Var): <i>numero_de_clases</i>	73	<i>ParseNoun</i> (Rut): <i>InterpretarNombre</i>	88
<i>NUMBER_PE</i> (Cons): <i>NUMERO_PE</i>	102	<i>ParseNumber</i> (Rut): <i>InterpretarNumero</i>	88
<i>NUMBER_TASKS</i> (Cons): <i>NUMERO_TAREAS</i>		<i>Parser.h</i> (Fich): <i>EParser.h</i>	9
97		<i>Parser_parse</i> (Rut): <i>Parser_parse</i>	93
<i>NUMBER_TOKEN</i> (Cons): <i>TOKEN_NUMERO</i>		<i>parser_action</i> (Var): <i>accion_parser</i>	71
110		<i>parser_inflection</i> (Var): <i>parser_inflexion</i>	74
<i>NumberWord</i> (Rut): <i>PalabraNumero</i>	93	<i>parser_one</i> (Var): <i>parser_uno</i>	74
<i>nw_obj</i> (Obj): <i>obj_no</i>	78	<i>parser_trace</i> (Var): <i>parser_trace</i>	74
<i>nw_to</i> (Prop): <i>al_no</i>	18	<i>parser_two</i> (Var): <i>parser_dos</i>	74
		<i>ParserError</i> (Rut): <i>ErrorParser</i>	88
O		<i>parterm.h</i> (Fich): <i>Eparterm.h</i>	9
<i>OBJECT_SCORE</i> (Cons): <i>PUNTOS_OBJETO</i>	97	<i>ParseToken</i> (Rut): <i>InterpretarToken</i>	92
<i>ObjectIsUntouchable</i> (Rut): <i>ObjetoEsIntocable</i>	85,	<i>PARSING_REASON</i> (Cons): <i>RAZON_PARSING</i>	
93		108	
<i>Objects</i> (Acc): <i>Objetos</i>	40	<i>PARTINV_BIT</i> (Cons): <i>INFOPARCIAL_BIT</i>	99
<i>Objects1Sub</i> (Rut): <i>Objetos1Sub</i>	93	<i>pcount</i> (Var): <i>contadorp</i>	71
		<i>pcount2</i> (Var): <i>contadorp2</i>	71

PKEY1__KY (Cons): TECLAANT1__KY 107	R
PKEY2__KY (Cons): TECLAANT2__KY 107	R_Process (Rut): R_Process 93
PKEY__TX (Cons): TECLAANT__TX 107	random (Rut): random 81
placed_in_flag (Var): bandera_puesto_en 71	RandomEntry (Rut): ElementoCualquiera 83
PlaceInScope (Rut): PonerAlAlcance 85	react_after (Prop): reaccionar_despues 32
Places (Acc): Lugares 40	REACT_AFTER_REASON (Cons): RA- ZON_REACCIONAR_DESPUES ... 108
Places1Sub (Rut): Lugares1Sub 92	react_before (Prop): reaccionar_antes 31
places_score (Var): puntos_sitios 75	REACT_BEFORE_REASON (Cons): RA- ZON_REACCIONAR_ANTES 108
player (Var): jugador 68	real_location (Var): localizacion_real 68
PlayerTo (Rut): JugadorA 84	reason_code (Var): codigo_razon 71
plural (Prop): plural 30	Receive (Acc): Recibir 65
PLURAL_BIT (Cons): PLURAL_BIT 109	receive_action (Var): accion_recibir 71
PluralFound (Acc): HalladoPlural 64	RECURSE_BIT (Cons): RECURSIVO_BIT ... 100
pluralname (Atr): nombreplural 14	Refers (Rut): SeRefiere 94
Pray (Acc): Rezar 47	Release (Cons): Release 97
Predictable (Acc): Predecible 63	Remove (Acc): Sacar 60
PrefaceByArticle (Rut): PonerArticuloDelante .. 93	REPARSE_CODE (Cons): CODIGO_REPARSE 98
PREPOSITION_TT (Cons): TT_PREPOSICION 111	ResetDescriptors (Rut): BorrarDescriptores 90
PrepositionChain (Rut): CadenaDePreposiciones 90	ResetVagueWords (Rut): ReiniciarPalabrasVagas 93
pretty_flag (Var): bandera_guapo 71	Restart (Acc): Reiniciar 40
Print_PName (Rut): Print_PName 81	RESTART__WD (Cons): REINICIAR__WD ... 106
Print_Spaces (Rut): Imprimir_Espacios 92	Restore (Acc): Restaurar 41
print_player_flag (Var): bandera_imprime_jugador 71	RESTORE__WD (Cons): RECUPERAR__WD 106
Print_ScL (Rut): Print_ScL 93	ReviseMulti (Rut): RevisarMulti 93
PrintCommand (Rut): ImprimirComando 92	RKEY__TX (Cons): RKEY__TX 106
PrintOrRun (Rut): ImprimirOEjecutar 84	ROOM_SCORE (Cons): PUNTOS_LUGAR ... 97
PrintRank (Rut): ImprimirRango 88	ROUTINE_FILTER_TT (Cons): TT_FILTRO Rutina 111
PrintShortName (Rut): PrintShortName 86	RoutinesOff (Acc): DesactivarRutinas 63
PrintTaskName (Rut): ImprimirTareas 88	RoutinesOn (Acc): ActivarRutinas 63
PrintVerb (Rut): ImprimirVerbo 88	Rub (Acc): Frotar 43
Prompt (Acc): Prompt 65	RunLife (Rut): EjecutarVida 91
pronoun_obj (Var): objeto_pronombre 74	RunRoutines (Rut): EjecutarRutinas 91
pronoun_word (Var): palabra_pronombre 74	RunTimeError (Rut): ErrorDeEjecucion 91
PronounNotice (Rut): ActualizarPronombre 81	S
Pronouns (Acc): Pronombres 40	s_obj (Obj): obj_s 78
PronounValue (Rut): ValorDelPronombre 87	s_to (Prop): al_s 18
proper (Atr): propio 14	SACK_OBJECT (Cons): OBJETO_SACO 97
PSN__ (Rut): PSN__ 93	Save (Acc): Salvar 41
Pull (Acc): Tirar 48	saved_oops (Var): eepa_guardado 72
Push (Acc): Empujar 42	SayWhatsOn (Rut): DecirQueHaySobre 90
PushDir (Acc): EmpujarDir 43	scenery (Atr): escenario 12
PutOn (Acc): PonerSobre 58	SCENERY_PE (Cons): ESCENARIO_PE 101
Q	Scope (Acc): Alcance 63
QKEY1__KY (Cons): QKEY1__KY 106	scope_error (Var): error_alcance 72
QKEY1__TX (Cons): QKEY1__TX 106	scope_reason (Var): razon_alcance 75
QKEY2__KY (Cons): QKEY2__KY 106	scope_stage (Var): estadio_alcance 72
QKEY2__TX (Cons): QKEY2__TX 106	scope_token (Var): token_alcance 75
Quit (Acc): Finalizar 39	SCOPE_TT (Cons): TT_ALCANCE 111
QUIT1__WD (Cons): TERMINAR1__WD 107	ScopeCeiling (Rut): TopeAlcanzable 94
QUIT2__WD (Cons): TERMINAR2__WD 107	ScopeWithin (Rut): DentroDelAlcance 83
	ScopeWithin_O (Rut): DentroDelAlcance_O ... 90

<i>Score</i> (Acc): Puntuacion	40	<i>sw_obj</i> (Obj): obj_so	78
<i>score</i> (Var): puntuacion	70	<i>sw_to</i> (Prop): al_so	18
<i>SCORE__TX</i> (Cons): PUNTUACION__TX	106	<i>Swim</i> (Acc): Nadar	45
<i>ScoreArrival</i> (Rut): PuntuacionLlegada	93	<i>Swing</i> (Acc): Columpiar	42
<i>scored</i> (Atr): valepuntos	15	<i>switchable</i> (Atr): conmutable	12
<i>ScoreMatchL</i> (Rut): PuntuacionListaEncajes	93	<i>SwitchOff</i> (Acc): Apagar	48
<i>ScriptOff</i> (Acc): DesactivarTranscripcion	39	<i>SwitchOn</i> (Acc): Encender	52
<i>ScriptOn</i> (Acc): ActivarTranscripcion	39		
<i>se_obj</i> (Obj): obj_se	78	T	
<i>se_to</i> (Prop): al_se	18	<i>Take</i> (Acc): Coger	49
<i>Search</i> (Acc): BuscarEn	49	<i>take_all_rule</i> (Var): regla_coger_todo	75
<i>SearchScope</i> (Rut): BuscarEnAlcance	90	<i>talkable</i> (Atr): hablable	13
<i>second</i> (Var): otro	69	<i>TALKING_REASON</i> (Cons): RAZON_HABLAR	108
<i>selfobj</i> (Obj): objjugador	78	<i>task_scores</i> (Var): puntuacion_tareas	70
<i>Serial</i> (Cons): Serial	97	<i>TASKS_PROVIDED</i> (Cons): HAY_TAREAS	96
<i>Set</i> (Acc): Fijar	43	<i>Taste</i> (Acc): Probar	46
<i>SetPronoun</i> (Rut): FijarPronombre	84	<i>Tell</i> (Acc): Hablar	44
<i>SetTime</i> (Rut): PonerLaHora	85	<i>TERSE_BIT</i> (Cons): BREVE_BIT	99
<i>SetTo</i> (Acc): PonerA	46	<i>TestScope</i> (Rut): PruebaDeAlcance	86
<i>short_name</i> (Prop): nombre_corto	27	<i>TESTSCOPE_REASON</i> (Cons): RAZON_TESTALCANCE	108
<i>short_name_case</i> (Var): caso_nombre_corto	71	<i>THAT__TX</i> (Cons): ESO__TX	104
<i>short_name_indef</i> (Prop): nombre_corto_indef	28	<i>THAT_BIT</i> (Cons): ESO_BIT	109
<i>Show</i> (Acc): Mostrar	58	<i>the_time</i> (Var): la_hora	68
<i>Showobj</i> (Acc): MostrarObjeto	63	<i>thedark</i> (Obj): laoscuridad	77
<i>Showverb</i> (Acc): MostrarVerbo	63	<i>THEN1__WD</i> (Cons): DESPUES1__WD	103
<i>sibling</i> (Rut): sibling	81	<i>THEN2__WD</i> (Cons): DESPUES2__WD	104
<i>SIEMPRE_BIT</i> (Cons): SIEMPRE_BIT	100	<i>THEN3__WD</i> (Cons): DESPUES3__WD	104
<i>Sing</i> (Acc): Cantar	42	<i>TheSame</i> (Acc): ElMismo	64
<i>Sleep</i> (Acc): Dormir	42	<i>things_score</i> (Var): puntos_cosas	75
<i>sline1</i> (Var): lineaEstado1	73	<i>Think</i> (Acc): Pensar	46
<i>sline2</i> (Var): lineaEstado2	73	<i>THOSET__TX</i> (Cons): ESASC__TX	104
<i>Smell</i> (Acc): Oler	45	<i>ThrowAt</i> (Acc): Lanzar	44
<i>Sorry</i> (Acc): LoSiento	45	<i>ThrownAt</i> (Acc): RecibirLanzamiento	65
<i>SortOutList</i> (Rut): OrdenarLista	93	<i>Tie</i> (Acc): Atar	41
<i>SortTogether</i> (Rut): OrdenarJuntos	93	<i>TIME__TX</i> (Cons): HORA__TX	104
<i>Spanish.h</i> (Fich): Espanol.h	9	<i>time_left</i> (Prop): tiempo_restante	35
<i>SpanishG.h</i> (Fich): Gramatica.h	9	<i>time_out</i> (Prop): tiempo_agotado	34
<i>special_number</i> (Var): numero_especial	74	<i>time_rate</i> (Var): hora_freq	72
<i>special_number1</i> (Var): numero_especial1	74	<i>time_step</i> (Var): hora_incr	72
<i>special_number2</i> (Var): numero_especial2	74	<i>TimePasses</i> (Rut): PasaElTiempo	89
<i>SPECIAL_TOKEN</i> (Cons): TOKEN_ESPECIAL	110	<i>TimersOff</i> (Acc): DesactivarReloj	63
<i>special_word</i> (Var): palabra_especial	74	<i>TimersOn</i> (Acc): ActivarReloj	62
<i>Squeeze</i> (Acc): Retorcer	47	<i>token_filter</i> (Var): filtro_token	72
<i>standard_interpreter</i> (Var): interprete_estandar ..	72	<i>Tokenise__</i> (Rut): Tokenise__	94
<i>StartDaemon</i> (Rut): ArrancarDaemon	81	<i>TOOFEW_PE</i> (Cons): HAYPOCOS_PE	101
<i>StartTimer</i> (Rut): ArrancarReloj	82	<i>TOOLIT_PE</i> (Cons): MUYPOCO_PE	102
<i>static</i> (Atr): estatico	12	<i>toomany_flag</i> (Var): bandera_demasiados	71
<i>StopDaemon</i> (Rut): PararDaemon	85	<i>top_object</i> (Var): objeto_raiz	74
<i>StopTimer</i> (Rut): PararReloj	85	<i>TOPIC_TOKEN</i> (Cons): TOKEN_TEMA	111
<i>Story</i> (Cons): Historia	96	<i>Touch</i> (Acc): Tocar	48
<i>STRICT_MODE</i> (Cons): STRICT_MODE	97	<i>TraceAction</i> (Rut): TrazarAccion	94
<i>Strong</i> (Acc): Tacos	47	<i>Tracelevel</i> (Acc): NivelTraza	63
<i>STUCK_PE</i> (Cons): ATASCADO_PE	101	<i>TraceOff</i> (Acc): DesactivarTraza	63
<i>supporter</i> (Atr): soporte	15		

<i>TraceOn</i> (Acc): ActivarTraza	63	<i>WakeOther</i> (Acc): DespertarOtro	42
<i>transcript_mode</i> (Var): modo_transcripcion	73	<i>Wave</i> (Acc): Agitar	41
<i>Transfer</i> (Acc): Transferir	61	<i>WaveHands</i> (Acc): Gesticular	43
<i>transparent</i> (Atr): transparente	15	<i>Wear</i> (Acc): Vestir	62
<i>true</i> (Cons): true	98	<i>when_closed</i> (Prop): si_cerrado	33
<i>TryGivenObject</i> (Rut): IntentarEncajarObjeto	92	<i>when_off</i> (Prop): si_apagado	33
<i>TryNumber</i> (Rut): IntentarNumero	84	<i>when_on</i> (Prop): si_encendido	34
<i>Turn</i> (Acc): Girar	43	<i>when_open</i> (Prop): si_abierto	32
<i>turns</i> (Var): turnos	70	<i>WHICH__TX</i> (Cons): ELCUAL__TX	104
U		<i>WHOM__TX</i> (Cons): QUIEN__TX	106
<i>u_obj</i> (Obj): obj_arriba	78	<i>WillRekurs</i> (Rut): DeboBajarRekursivo	90
<i>u_to</i> (Prop): arriba	19	<i>with_key</i> (Prop): con_llave	20
<i>UNDO1__WD</i> (Cons): ANULAR1__WD	103	<i>WITHOUT_DIRECTIONS</i> (Cons): SIN_DIRECCIONES	97
<i>UNDO2__WD</i> (Cons): ANULAR2__WD	103	<i>wlf_indent</i> (Var): indentacion_eld	72
<i>UNDO3__WD</i> (Cons): ANULAR3__WD	103	<i>wn</i> (Var): np	69
<i>undo_flag</i> (Var): bandera_deshacer	71	<i>WordAddress</i> (Rut): DireccionDePalabra	83
<i>UnknownVerb</i> (Rut): VerboDesconocido	90	<i>WordInProperty</i> (Rut): PalabraEnPropiedad	85
<i>UNLIT_BIT</i> (Cons): APAGADO_BIT	108	<i>WordLength</i> (Rut): LongitudDePalabra	85
<i>Unlock</i> (Acc): QuitarCerrojo	59	<i>workflag</i> (Atr): banderaux	11
<i>UnpackGrammarLine</i> (Rut): DesempaquetarLinea- Gramatica	91	<i>WORKFLAG_BIT</i> (Cons): BANDERAUX_BIT	99
<i>UnsignedCompare</i> (Rut): CompararSinSigno	82	<i>worn</i> (Atr): puesto	15
<i>UPTO_PE</i> (Cons): HASTAQUI_PE	101	<i>WriteAfterEntry</i> (Rut): EscribeTrasElemento	91
<i>UserFilter</i> (Rut): FiltroUsuario	91	<i>WriteBeforeEntry</i> (Rut): EscribeAntesElemento	91
<i>usual_grammar_after</i> (Var): gramatica_normal_tras	72	<i>WriteListFrom</i> (Rut): EscribirListaDesde	84, 91
V		<i>WriteListR</i> (Rut): EscribirListaR	91
<i>VAGUE_PE</i> (Cons): PRONOM_PE	102	X	
<i>VagueGo</i> (Acc): IrAmbiguo	44	<i>x_scope_count</i> (Var): x_cuenta_ambito	75
<i>ValueOrRun</i> (Rut): ValorOEjecutar	87	<i>XAbstract</i> (Acc): XMover	63
<i>VERB_PE</i> (Cons): VERBO_PE	102	<i>xcommsdir</i> (Var): xcommsdir	75
<i>verb_word</i> (Var): palabra_verbo	74	<i>XObj</i> (Rut): XObj	94
<i>verb_wordnum</i> (Var): palabra_verbonum	74	<i>XPurloin</i> (Acc): XRobar	63
<i>Verblib.h</i> (Fich): Acciones.h	9	<i>XTestMove</i> (Rut): XCompruebaMover	94
<i>verblibm.h</i> (Fich): accionm.h	9	<i>XTree</i> (Acc): XArbol	63
<i>Verify</i> (Acc): Verificar	41	Y	
<i>visibility_ceiling</i> (Var): techo_de_visibilidad	75	<i>Yes</i> (Acc): Si	47
<i>visited</i> (Atr): visitado	16	<i>YES1__WD</i> (Cons): SI1__WD	106
W		<i>YES2__WD</i> (Cons): SI2__WD	107
<i>w_obj</i> (Obj): obj_o	78	<i>YES3__WD</i> (Cons): SI3__WD	107
<i>w_to</i> (Prop): al_o	18	<i>YesOrNo</i> (Rut): SiONo	86
<i>Wait</i> (Acc): Esperar	43	<i>YOURSELF__TX</i> (Cons): TUMISMO__TX	107
<i>Wake</i> (Acc): Despertarse	42	Z	
		<i>ZRegion</i> (Rut): ZRegion	87

10.3. Equivalencias Español - Inglés

Símbolos

(MCoge) (Rut): <i>(Nuevo)</i>	80	al_ne (Prop): <i>ne_to</i>	18
(MMCoge) (Rut): <i>(Nuevo)</i>	80	al_no (Prop): <i>nw_to</i>	18
(_El) (Rut): <i>(The)</i>	79	al_o (Prop): <i>w_to</i>	18
(_nombre_) (Rut): <i>(name)</i>	80	al_s (Prop): <i>s_to</i>	18
(al) (Rut): <i>(Nuevo)</i>	79	al_se (Prop): <i>se_to</i>	18
(coge) (Rut): <i>(Nuevo)</i>	79	al_so (Prop): <i>sw_to</i>	18
(del) (Rut): <i>(Nuevo)</i>	79	AlAlcance (Rut): <i>InScope</i>	87
(e) (Rut): <i>(Nuevo)</i>	79	Alcance (Acc): <i>Scope</i>	63
(el) (Rut): <i>(the)</i>	79	amodo_noposeido (Var): <i>onotheld_mode</i>	71
(es) (Rut): <i>(Nuevo)</i>	80	AnalizarToken (Rut): <i>AnalyseToken</i>	90
(esta) (Rut): <i>(Nuevo)</i>	80	ancho_elemento (Var): <i>item_width</i>	71
(lo) (Rut): <i>(Nuevo)</i>	80	AniadirMulti (Rut): <i>MultiAdd</i>	90
(n) (Rut): <i>(Nuevo)</i>	80	anidacion_menu (Var): <i>menu_nesting</i>	71
(numero) (Rut): <i>(number)</i>	80	ANIMA_PE (Cons): <i>ANIMA_PE</i>	101
(o) (Rut): <i>(Nuevo)</i>	80	animado (Atr): <i>animate</i>	11
(s) (Rut): <i>(Nuevo)</i>	80	AnotarLlegada (Rut): <i>NoteArrival</i>	90
(un) (Rut): <i>(a)</i>	80	AnotarObjetosObtenidos (Rut): <i>NoteObjectAcquisitions</i>	90

A

abajo (Prop): <i>d_to</i>	17	AntepasadoComun (Rut): <i>CommonAncestor</i>	90
abierto (Atr): <i>open</i>	11	antes (Prop): <i>before</i>	18
abrible (Atr): <i>openable</i>	11	AntesDelParsing (Rut): <i>BeforeParsing</i>	87
Abrir (Acc): <i>Open</i>	48	ANULAR1__WD (Cons): <i>UNDO1__WD</i>	103
accion (Var): <i>action</i>	67	ANULAR2__WD (Cons): <i>UNDO2__WD</i>	103
accion_invertida (Var): <i>action_reversed</i>	70	ANULAR3__WD (Cons): <i>UNDO3__WD</i>	103
accion_parser (Var): <i>parser_action</i>	71	Anuncio (Rut): <i>Banner</i>	90
accion_que_seria (Var): <i>action_to_be</i>	71	APAGADO_BIT (Cons): <i>UNLIT_BIT</i>	108
accion_recibir (Var): <i>receive_action</i>	71	Apagar (Acc): <i>SwitchOff</i>	48
Acciones.h (Fich): <i>VerbLib.h</i>	9	ArrancarDaemon (Rut): <i>StartDaemon</i>	81
accionm.h (Fich): <i>verbLibm.h</i>	9	ArrancarReloj (Rut): <i>StartTimer</i>	82
ActivarAcciones (Acc): <i>ActionsOn</i>	62	arriba (Prop): <i>u_to</i>	19
ActivarAcentos (Acc): <i>(Nuevo)</i>	62	articulo (Prop): <i>article</i>	19
ActivarComandos (Acc): <i>CommandsOn</i>	62	articulos (Prop): <i>articles</i>	19
ActivarNotificacion (Acc): <i>NotifyOn</i>	39	Atacar (Acc): <i>Attack</i>	41
ActivarReloj (Acc): <i>TimersOn</i>	62	Atar (Acc): <i>Tie</i>	41
ActivarRutinas (Acc): <i>RoutinesOn</i>	63	ATASCADO_PE (Cons): <i>STUCK_PE</i>	101
ActivarTranscripcion (Acc): <i>ScriptOn</i>	39	ausente (Atr): <i>absent</i>	11
ActivarTraza (Acc): <i>TraceOn</i>	63	AveriguarPreposicion (Rut): <i>(Nuevo)</i>	90
actor (Var): <i>actor</i>	67	AveriguarPrimeraPreposicion (Rut): <i>(Nuevo)</i>	90
ActualizarEstatus (Rut): <i>DisplayStatus</i>	90	aviso_avanzar (Var): <i>advance_warning</i>	71
ActualizarPronombre (Rut): <i>PronounNotice</i>	81		
adentro (Prop): <i>in_to</i>	17		
adjetivos (Prop): <i>(Nuevo)</i>	18		
ADMITIR_COMANDO_SALIDAS (Cons): <i>(Nuevo)</i>	95		
ADMITIR_INFINITIVOS (Cons): <i>(Nuevo)</i>	95		
afuera (Prop): <i>out_to</i>	18		
Agitar (Acc): <i>Wave</i>	41		
AjustarLuz (Rut): <i>AdjustLight</i>	90		
al_e (Prop): <i>e_to</i>	18		
al_n (Prop): <i>n_to</i>	18		

B

Bajar (Acc): <i>(Nuevo)</i>	48
bandera_debug (Var): <i>debug_flag</i>	71
bandera_demasiados (Var): <i>toomany_flag</i>	71
bandera_deshacer (Var): <i>undo_flag</i>	71
bandera_guapo (Var): <i>pretty_flag</i>	71
bandera_imprime_jugador (Var): <i>print_player_flag</i>	71
bandera_paa (Var): <i>ats_flag</i>	71
bandera_puesto_en (Var): <i>placed_in_flag</i>	71
banderafin (Var): <i>deadflag</i>	67
banderaluz (Var): <i>lightflag</i>	71
banderamulti (Var): <i>multiflag</i>	71

banderaux (Atr): <i>workflag</i>	11	Cortar (Acc): <i>Cut</i>	42
BANDERAUX_BIT (Cons): <i>WORKFLAG_BIT</i>	99	Crear__IN (Cons): <i>Make__PN</i>	109
Beber (Acc): <i>Drink</i>	41	Crear__IR (Cons): <i>Make__PR</i>	109
Besar (Acc): <i>Kiss</i>	41	Curiosidades (Rut): <i>Amusing</i>	87
best_tipoerror (Var): <i>best_etype</i>	71	CURIOSIDADES__WD (Cons): <i>AMUSING__WD</i>	103
BorrarDescriptores (Rut): <i>ResetDescriptors</i>	90		
BREVE_BIT (Cons): <i>TERSE_BIT</i>	99		
brujula (Obj): <i>compass</i>	77		
BucleAlcance (Rut): <i>LoopOverScope</i>	82		
buffer (Var): <i>buffer</i>	71		
buffer2 (Var): <i>buffer2</i>	71		
buffer3 (Var): <i>buffer3</i>	71		
bufferaux (Var): <i>(Nuevo)</i>	71		
BuscarEn (Acc): <i>Search</i>	49		
BuscarEnAlcance (Rut): <i>SearchScope</i>	90		
BuscarEnDiccionario (Rut): <i>DictionaryLookup</i> ..	82		
BuscarEntreVerbosIrregulares (Rut): <i>(Nuevo)</i> ...	90		
C		D	
cada_turno (Prop): <i>each_turn</i>	20	daemon (Prop): <i>daemon</i>	20
CadenaDePreposiciones (Rut): <i>PrepositionChain</i> ..	90	Dar (Acc): <i>Give</i>	50
CambiarDefecto (Rut): <i>ChangeDefault</i>	82	dat1 (Var): <i>inp1</i>	71
CambiarJugador (Rut): <i>ChangePlayer</i>	82	dat2 (Var): <i>inp2</i>	72
CaminarAOscuras (Rut): <i>DarkToDark</i>	87	DE1__WD (Cons): <i>OF1__WD</i>	103
Cantar (Acc): <i>Sing</i>	42	DE2__WD (Cons): <i>OF2__WD</i>	103
cantidad (Prop): <i>number</i>	20	DE3__WD (Cons): <i>OF3__WD</i>	103
capacidad (Prop): <i>capacity</i>	20	DE4__WD (Cons): <i>OF4__WD</i>	103
caso_nombre_corto (Var): <i>short_name_case</i>	71	DeboBajarRecursivo (Rut): <i>WillRecurs</i>	90
CasosLenguaje (Cons): <i>LanguageCases</i>	109	DEBUG (Cons): <i>DEBUG</i>	95
CDefArt (Rut): <i>CDefArt</i>	82	DecirQueHaySobre (Rut): <i>SayWhatsOn</i>	90
Cerrar (Acc): <i>Close</i>	49	deducedesde (Var): <i>inferfrom</i>	72
cerrojo (Atr): <i>lockable</i>	11	DefArt (Rut): <i>DefArt</i>	83
cerrojoechoado (Atr): <i>locked</i>	12	DEFINIDO_BIT (Cons): <i>DEFART_BIT</i>	99
child (Rut): <i>child</i>	80	DEFINIDO_PK (Cons): <i>DEFART_PK</i>	109
children (Rut): <i>children</i>	81	Dejar (Acc): <i>Drop</i>	51
CientosEspanol (Rut): <i>(Nuevo)</i>	90	DejarSalir (Acc): <i>LetGo</i>	63
codigo_razon (Var): <i>reason_code</i>	71	DentroDelAlcance (Rut): <i>ScopeWithin</i>	83
CODIGO_REPARSE (Cons): <i>REPARSE_CODE</i> ..	98	DentroDelAlcance_O (Rut): <i>ScopeWithin_O</i> ...	90
Coger (Acc): <i>Take</i>	49	DepurarAccion (Rut): <i>DebugAction</i>	90
Columpiar (Acc): <i>Swing</i>	42	DepurarAtributo (Rut): <i>DebugAttribute</i>	91
Comer (Acc): <i>Eat</i>	50	DepurarLineaGramatica (Rut): <i>DebugGrammarLine</i> ..	91
comestible (Atr): <i>edible</i>	12		
CompararSinSigno (Rut): <i>UnsignedCompare</i> ...	82	DepurarParametro (Rut): <i>DebugParameter</i>	91
Comprar (Acc): <i>Buy</i>	42	DepurarToken (Rut): <i>DebugToken</i>	91
con_llave (Prop): <i>with_key</i>	20	DesactivarAcciones (Acc): <i>ActionsOff</i>	63
conmutable (Atr): <i>switchable</i>	12	DesactivarAcentos (Acc): <i>(Nuevo)</i>	63
Conseguido (Rut): <i>Achieved</i>	82	DesactivarComandos (Acc): <i>CommandsOff</i>	63
Consultar (Acc): <i>Consult</i>	42	DesactivarNotificacion (Acc): <i>NotifyOff</i>	39
consultar_desde (Var): <i>consult_from</i>	71	DesactivarReloj (Acc): <i>TimersOff</i>	63
consultar_num_palabras (Var): <i>consult_words</i> ...	71	DesactivarRutinas (Acc): <i>RoutinesOff</i>	63
contadorp (Var): <i>pcount</i>	71	DesactivarTranscripcion (Acc): <i>ScriptOff</i>	39
contadorp2 (Var): <i>pcount2</i>	71	DesactivarTraza (Acc): <i>TraceOff</i>	63
ContieneIndirectamente (Rut): <i>IndirectlyContains</i> ..	90	describir (Prop): <i>describe</i>	20
		descripcion (Prop): <i>description</i>	21
CopiarBuffer (Rut): <i>CopyBuffer</i>	90	descripcion_dentro (Prop): <i>inside_description</i> ...	21
		Descriptores (Rut): <i>Descriptors</i>	91
		DesempaquetarLineaGramatica (Rut): <i>Unpack-GrammarLine</i>	91
		DespertarOtro (Acc): <i>WakeOther</i>	42
		Despertarse (Acc): <i>Wake</i>	42
		despues (Prop): <i>after</i>	21
		DESPUES11_WD (Cons): <i>(Nuevo)</i>	103
		DESPUES1__WD (Cons): <i>THEN1__WD</i>	103
		DESPUES21_WD (Cons): <i>(Nuevo)</i>	104
		DESPUES2__WD (Cons): <i>THEN2__WD</i>	104
		DESPUES31_WD (Cons): <i>(Nuevo)</i>	104

DESPUES3__WD (Cons): <i>THEN3__WD</i>	104	ESO_BIT (Cons): <i>THAT_BIT</i>	109
Desvestir (Acc): <i>Disrobe</i>	51	Espanol.h (Fich): <i>Spanish.h</i>	9
Dialecto (Acc): <i>(Nuevo)</i>	39	ESPAÑOL_BIT (Cons): <i>ENGLISH_BIT</i>	99
dialecto_sudamericano (Var): <i>(Nuevo)</i>	68	EspanolAInformes (Rut): <i>(Nuevo)</i>	91
DialectoCast (Acc): <i>(Nuevo)</i>	39	Esperar (Acc): <i>Wait</i>	43
DialectoSud (Acc): <i>(Nuevo)</i>	39	esta_en (Prop): <i>found_in</i>	22
DibujarLineaEstado (Rut): <i>DrawStatusLine</i>	83	estadio_alcance (Var): <i>scope_stage</i>	72
DigitoEspanol (Rut): <i>(Nuevo)</i>	91	estatico (Atr): <i>static</i>	12
DirDicc__Num (Rut): <i>Dword__No</i>	91	estilo_ac (Var): <i>c_style</i>	72
direcc_puerta (Prop): <i>door_dir</i>	22	estilo_inventario (Var): <i>inventory_style</i>	72
DireccionDePalabra (Rut): <i>WordAddress</i>	83	etapa_inventario (Var): <i>inventory_stage</i>	68
DoMenu (Rut): <i>DoMenu</i>	83	Examinar (Acc): <i>Examine</i>	52
DominioNombre (Rut): <i>NounDomain</i>	83	Excavar (Acc): <i>Dig</i>	43
Dormir (Acc): <i>Sleep</i>	42	EXCEPTO_PE (Cons): <i>EXCEPT_PE</i>	101

E

EcharCerrojo (Acc): <i>Lock</i>	51
eeпа_desde (Var): <i>oops_from</i>	72
eeпа_guardado (Var): <i>saved_oops</i>	72
EfectuarAccionesAlcance (Rut): <i>DoScopeAction</i>	91
EjecutarRutinas (Rut): <i>RunRoutines</i>	91
EjecutarVida (Rut): <i>RunLife</i>	91
ELCUAL__TX (Cons): <i>WHICH__TX</i>	104
ElegirObjetos (Rut): <i>ChooseObjects</i>	88
elemento_menu (Var): <i>menu_item</i>	72
ElementoCualquiera (Rut): <i>RandomEntry</i>	83
EligeObjetos (Rut): <i>(Nuevo)</i>	91
EliminarDuplicados (Rut): <i>(Nuevo)</i>	91
ElMismo (Acc): <i>TheSame</i>	64
elsiguiente (Var): <i>lookahead</i>	72
Empujar (Acc): <i>Push</i>	42
EmpujarDir (Acc): <i>PushDir</i>	43
encajado_desde (Var): <i>match_from</i>	72
Encender (Acc): <i>SwitchOn</i>	52
encendido (Atr): <i>on</i>	12
ENCENDIDO_BIT (Cons): <i>LIT_BIT</i>	109
EnglishNumber (Rut): <i>EnglishNumber</i>	83, 91
enlazlpa.h (Fich): <i>linklpa.h</i>	10
enlazlv.h (Fich): <i>linklv.h</i>	10
entrable (Atr): <i>enterable</i>	12
Entrar (Acc): <i>Enter</i>	52
EParser.h (Fich): <i>Parser.h</i>	9
Eparserm.h (Fich): <i>parserm.h</i>	9
error_alcance (Var): <i>scope_error</i>	72
ErrorDeEjecucion (Rut): <i>RunTimeError</i>	91
ErrorParser (Rut): <i>ParserError</i>	88
ESASC__TX (Cons): <i>THOSET__TX</i>	104
escenario (Atr): <i>scenery</i>	12
ESCENARIO_PE (Cons): <i>SCENERY_PE</i>	101
EscribeAntesElemento (Rut): <i>WriteBeforeEntry</i>	91
EscribeListaR (Rut): <i>WriteListR</i>	91
EscribeTrasElemento (Rut): <i>WriteAfterEntry</i>	91
EscribirListaDesde (Rut): <i>WriteListFrom</i>	84, 91
Escuchar (Acc): <i>Listen</i>	43
ESO__TX (Cons): <i>THAT__TX</i>	104

F

Facilitar.h (Fich): <i>(Nuevo)</i>	10
false (Cons): <i>false</i>	98
femenino (Atr): <i>female</i>	13
Fijar (Acc): <i>Set</i>	43
FijarPronombre (Rut): <i>SetPronoun</i>	84
FiltrarMulti (Rut): <i>MultiFilter</i>	91
filtro_token (Var): <i>token_filter</i>	72
FiltroUsuario (Rut): <i>UserFilter</i>	91
Finalizar (Acc): <i>Quit</i>	39
Frotar (Acc): <i>Rub</i>	43

G

G_FEMENINO (Cons): <i>(Nuevo)</i>	98
G_MASCULINO (Cons): <i>(Nuevo)</i>	98
G_PLURAL (Cons): <i>(Nuevo)</i>	98
general (Atr): <i>general</i>	13
genero (Prop): <i>(Nuevo)</i>	23
Gesticular (Acc): <i>WaveHands</i>	43
Girar (Acc): <i>Turn</i>	43
gramatica (Prop): <i>grammar</i>	24
Gramatica.h (Fich): <i>SpanishG.h</i>	9
gramatica_normal_tras (Var): <i>usual_grammar_after</i>	72
Grammar__Version (Cons): <i>Grammar__Version</i>	109

H

hablable (Atr): <i>talkable</i>	13
Hablar (Acc): <i>Tell</i>	44
HacerEncaje (Rut): <i>MakeMatch</i>	91
HalladoPlural (Acc): <i>PluralFound</i>	64
HASTAQUI_PE (Cons): <i>UPTO_PE</i>	101
HAY2__TX (Cons): <i>IS2__TX</i>	104
HAY__TX (Cons): <i>IS__TX</i>	104
HAY_BIT (Cons): <i>ISARE_BIT</i>	99
HAY_CURIOSIDADES (Cons): <i>AMUS- ING_PROVIDED</i>	95
HAY_TAREAS (Cons): <i>TASKS_PROVIDED</i>	96
HayLuz (Rut): <i>OffersLight</i>	84
HAYP2__TX (Cons): <i>ARE2__TX</i>	104

HAYP__TX (Cons): ARE__TX	104	IniciarPregunta (Rut): (Nuevo)	92
HAYPOCOS_PE (Cons): TOOFEW_PE	101	InsertarLetraBuffer (Rut): (Nuevo)	92
herobj (Var): herobj	72	IntentarCogerObjeto (Rut): AttemptToTakeObject	92
himobj (Var): himobj	72	IntentarEncajarObjeto (Rut): TryGivenObject	92
Historia (Cons): Story	96	IntentarNumero (Rut): TryNumber	84
HORA__TX (Cons): TIME__TX	104	InterpretarNombre (Rut): ParseNoun	88
hora_freq (Var): time_rate	72	InterpretarNumero (Rut): ParseNumber	88
hora_incr (Var): time_step	72	InterpretarToken (Rut): ParseToken	92

I

Identicos (Rut): Identical	91
IdiomaAInformes (Rut): LanguageToInformese ..	91
IdiomaContraccion (Rut): LanguageContraction ..	92
IdiomaContracciones (Cons): LanguageContractionForms	109
IdiomaDirecciones (Rut): LanguageDirection ..	92
IdiomaEsVerbo (Rut): LanguageIsVerb	92
IdiomaGeneroAnimado (Cons): LanguageAnimateGender	109
IdiomaGeneroInanimado (Cons): LanguageInanimateGender	109
IdiomaHoraDelDia (Rut): LanguageTimeOfDay ..	92
IdiomaImprimirComando (Rut): LanguagePrintCommand	92
IdiomaImprimirNombreCorto (Rut): LanguagePrintShortName	92
IdiomaNumero (Rut): LanguageNumber	92
IdiomaPreguntarPrep (Rut): (Nuevo)	92
IdiomaVerbo (Rut): LanguageVerb	92
imperativo (Prop): (Nuevo)	25
ImpNumAlin (Rut): PANum	92
Imprimir_Espacios (Rut): Print_Spaces	92
ImprimirComando (Rut): PrintCommand	92
ImprimirIrregular (Rut): (Nuevo)	92
ImprimirOExecutar (Rut): PrintOrRun	84
ImprimirRango (Rut): PrintRank	88
ImprimirTareas (Rut): PrintTaskName	88
ImprimirVerbo (Rut): PrintVerb	88
indef_casos (Var): indef_cases	72
indef_esperado (Var): indef_wanted	72
indef_numero_en (Var): indef_nspec_at	72
indef_posibambig (Var): indef_possambig	72
indef_propietario (Var): indef_owner	72
indef_suponer_p (Var): indef_guess_p	72
indef_tipo (Var): indef_type	72
IndefArt (Rut): InDefArt	84
indentacion_eld (Var): wlf_indent	72
INDENTAR_BIT (Cons): INDENT_BIT	99
indirect (Rut): indirect	81
INFIX (Cons): INFIX	96
infixe.h (Fich): infix.h	10
INFOPARCIAL_BIT (Cons): PARTINV_BIT ..	99
INFOTOTAL_BIT (Cons): FULLINV_BIT	100
inicial (Prop): initial	25
Inicializar (Rut): Initialise	88

J

JUGADAS__TX (Cons): MOVES__TX	104
jugador (Var): player	68
JugadorA (Rut): PlayerTo	84
just_undone (Var): just_undone	73

K

KKFINAL_PE (Cons): JUNKAFTER_PE	101
---------------------------------------	-----

L

la_hora (Var): the_time	68
Lanzar (Acc): ThrowAt	44
laoscuridad (Obj): thedark	77
lastdesc (Var): lastdesc	73
LeerComandos (Acc): CommandsRead	63
LibreriaInform (Obj): InformLibrary	77
lineaEstado1 (Var): sline1	73
lineaEstado2 (Var): sline2	73
ListaMiscelanea (Acc): ListMiscellany	64
listando_junto (Var): listing_together	73
listar_juntos (Prop): list_together	26
ListarIguales (Rut): ListEqual	92
listarse (Prop): invent	26
Llenar (Acc): Fill	44
LLEVAR_MAX (Cons): MAX_CARRIED	96
Local (Rut): Locale	84
localizacion (Var): location	68
localizacion_actor (Var): actors_location	73
localizacion_real (Var): real_location	68
long_encajado (Var): match_length	73
LongitudDePalabra (Rut): WordLength	85
LoSiento (Acc): Sorry	45
LowKey_Menu (Rut): LowKey_Menu	92
Lugares (Acc): Places	40
Lugares1Sub (Rut): Places1Sub	92
LugarNuevo (Rut): NewRoom	89

luz (Atr): <i>light</i>	13	Nadar (Acc): <i>Swim</i>	45
M		neutro (Atr): <i>neuter</i>	14
M__L (Rut): <i>L__M</i>	92	nextbest_tipoerror (Var): <i>nextbest_etype</i>	73
M__L (Rut): <i>L__M</i>	92	NivelTraza (Acc): <i>Tracelevel</i>	63
Main (Rut): <i>Main</i>	92	No (Acc): <i>No</i>	45
mant_np (Var): <i>hb_wn</i>	73	NO1__WD (Cons): <i>NO1__WD</i>	104
MasAlla (Rut): <i>AfterLife</i>	89	NO2__WD (Cons): <i>NO2__WD</i>	104
masculino (Atr): <i>male</i>	13	NO3__WD (Cons): <i>NO3__WD</i>	105
MAX_RELOJES (Cons): <i>MAX_TIMERS</i>	96	no_deducir (Var): <i>dont_infer</i>	73
MejorIntuicion (Rut): <i>BestGuess</i>	92	no_interpretar (Var): <i>no_parse</i>	69
MensajeMuerte (Rut): <i>DeathMessage</i>	89	NO_LUGARES (Cons): <i>NO_PLACES</i>	96
Mensajes.h (Fich): <i>(Nuevo)</i>	10	no_poseido (Var): <i>not_holding</i>	73
MensajesLibreria (Obj): <i>LibraryMessages</i>	77	no_puedes_ir (Prop): <i>cant_go</i>	28
meta (Var): <i>meta</i>	73	NO_PUNTUACION (Cons): <i>(Nuevo)</i>	96
metaclass (Rut): <i>metaclass</i>	81	NoComprendido (Acc): <i>NotUnderstood</i>	64
Meter (Acc): <i>Insert</i>	55	NoLoVeo (Rut): <i>CantSee</i>	93
Meterse (Acc): <i>GoIn</i>	56	nombre (Prop): <i>name</i>	27
MI_BIT (Cons): <i>MY_BIT</i>	109	nombre_corto (Prop): <i>short_name</i>	27
Mirar (Acc): <i>Look</i>	56	nombre_corto_indef (Prop): <i>short_name_indef</i>	28
MirarDebajo (Acc): <i>LookUnder</i>	45	nombre_elemento (Var): <i>item_name</i>	73
MirarHaciaSub (Rut): <i>(Nuevo)</i>	89	nombre_f (Prop): <i>(Nuevo)</i>	28
Miscelanea (Acc): <i>Miscellany</i>	64	nombre_fp (Prop): <i>(Nuevo)</i>	28
ml_n (Var): <i>lm_n</i>	73	nombre_m (Prop): <i>(Nuevo)</i>	28
ml_o (Var): <i>lm_o</i>	73	nombre_mp (Prop): <i>(Nuevo)</i>	28
MLIdioma (Rut): <i>LanguageLM</i>	92	nombreprural (Atr): <i>pluralname</i>	14
MMULTI_PE (Cons): <i>MMULTI_PE</i>	101	nombreado (Atr): <i>(Nuevo)</i>	14
modo_indef (Var): <i>indef_mode</i>	73	NOPUEDESIR_TX (Cons): <i>CANTGO_TX</i>	105
modo_mantenido (Var): <i>held_back_mode</i>	73	NoSoportaAlJugador (Rut): <i>NotSupportingTheP-layer</i>	93
modo_multi (Var): <i>multi_mode</i>	73	nothing (Cons): <i>nothing</i>	98
modo_noposeido (Var): <i>noheld_mode</i>	73	NOTIENES_PE (Cons): <i>NOTHELD_PE</i>	102
modo_notificar (Var): <i>notify_mode</i>	69	NotificarLaPuntuacion (Rut): <i>NotifyTheScore</i>	93
modo_transcripcion (Var): <i>transcript_mode</i>	73	NOVEO_PE (Cons): <i>CANTSEE_PE</i>	102
ModoM1 (Acc): <i>LMode1</i>	40	np (Var): <i>wn</i>	69
ModoM2 (Acc): <i>LMode2</i>	40	nsns (Var): <i>nsns</i>	73
ModoM3 (Acc): <i>LMode3</i>	40	NUEVALINEA_BIT (Cons): <i>NEWLINE_BIT</i>	100
modomirar (Var): <i>lookmode</i>	73	Nuevos identificadores	
Mostrar (Acc): <i>Show</i>	58	(MCoge) (Rut)	80
MostrarObjeto (Acc): <i>Showobj</i>	63	(MMCoge) (Rut)	80
MostrarVerbo (Acc): <i>Showverb</i>	63	(al) (Rut)	79
MoverJugador (Rut): <i>MovePlayer</i>	92	(coge) (Rut)	79
MoverObjetosFlotantes (Rut): <i>MoveFloatingObjects</i>	92	(del) (Rut)	79
MoverPalabra (Rut): <i>MoveWord</i>	93	(e) (Rut)	79
movido (Atr): <i>moved</i>	13	(es) (Rut)	80
Msg1P.h (Fich): <i>(Nuevo)</i>	10	(esta) (Rut)	80
multi_contexto (Var): <i>multi_context</i>	73	(lo) (Rut)	80
multi_esperado (Var): <i>multi_wanted</i>	73	(n) (Rut)	80
multi_hallado (Var): <i>multi_had</i>	73	(o) (Rut)	80
MULTI_PE (Cons): <i>MULTI_PE</i>	102	(s) (Rut)	80
MUYPOCO_PE (Cons): <i>TOOLIT_PE</i>	102	ActivarAcentos (Acc)	62
N		adjetivos (Prop)	18
NADA_TX (Cons): <i>NOTHING_TX</i>	104	ADMITIR_COMANDO_SALIDAS (Cons)	95
NADA_PE (Cons): <i>NOTHING_PE</i>	102	ADMITIR_INFINITIVOS (Cons)	95
		AveriguarPreposicion (Rut)	90
		AveriguarPrimeraPreposicion (Rut)	90

Bajar (Acc)	48	numero_interpretado (Var): <i>parsed_number</i>	74
bufferaux (Var)	71	NUMERO_PE (Cons): <i>NUMBER_PE</i>	102
BuscarEntreVerbosIrregulares (Rut)	90	NUMERO_TAREAS (Cons): <i>NUMBER_TASKS</i>	
CientosEspanol (Rut)	90		97
DesactivarAcentos (Acc)	63	NumSerieLib (Cons): <i>LibSerial</i>	109
DESPUES11_WD (Cons)	103		
DESPUES21_WD (Cons)	104	O	
DESPUES31_WD (Cons)	104	O__TX (Cons): <i>OR__TX</i>	105
Dialecto (Acc)	39	obj_abajo (Obj): <i>d_obj</i>	78
dialecto_sudamericano (Var)	68	obj_afuera (Obj): <i>out_obj</i>	78
DialectoCast (Acc)	39	obj_arriba (Obj): <i>u_obj</i>	78
DialectoSud (Acc)	39	obj_dentro (Obj): <i>in_obj</i>	78
DigitoEspanol (Rut)	91	obj_e (Obj): <i>e_obj</i>	78
EligeObjetos (Rut)	91	obj_n (Obj): <i>n_obj</i>	78
EliminarDuplicados (Rut)	91	obj_ne (Obj): <i>ne_obj</i>	78
EspanolAInformes (Rut)	91	obj_no (Obj): <i>nw_obj</i>	78
Facilitar.h (Fich)	10	obj_o (Obj): <i>w_obj</i>	78
G_FEMENINO (Cons)	98	obj_s (Obj): <i>s_obj</i>	78
G_MASCULINO (Cons)	98	obj_se (Obj): <i>se_obj</i>	78
G_PLURAL (Cons)	98	obj_so (Obj): <i>sw_obj</i>	78
genero (Prop)	23	objeto_pronombre (Var): <i>pronoun_obj</i>	74
IdiomaPreguntarPrep (Rut)	92	objeto_raiz (Var): <i>top_object</i>	74
imperativo (Prop)	25	OBJETO_SACO (Cons): <i>SACK_OBJECT</i>	97
ImprimirIrregular (Rut)	92	ObjetoEnAlcancePorAlgo (Rut): <i>ObjectScoped-</i>	
IniciarPregunta (Rut)	92	<i>BySomething</i>	93
InsertarLetraBuffer (Rut)	92	ObjetoEsIntocable (Rut): <i>ObjectIsUntouchable</i> .	85,
irrelevante (Prop)	26		93
Mensajes.h (Fich)	10	Objetos (Acc): <i>Objects</i>	40
MirarHaciaSub (Rut)	89	Objetos1Sub (Rut): <i>Objects1Sub</i>	93
Msg1P.h (Fich)	10	objjugador (Obj): <i>selfobj</i>	78
NO_PUNTUACION (Cons)	96	ObtenerGenero (Rut): <i>GetGender</i>	93
nombre_f (Prop)	28	ObtenerGNADeObjeto (Rut): <i>GetGNAOfObject</i>	93
nombre_fp (Prop)	28	OCULTAR_BIT (Cons): <i>CONCEAL_BIT</i>	100
nombre_m (Prop)	28	oculto (Atr): <i>concealed</i>	14
nombre_mp (Prop)	28	old_herobj (Var): <i>old_herobj</i>	74
nombreado (Atr)	14	old_himobj (Var): <i>old_himobj</i>	74
parseaux (Var)	74	old_itobj (Var): <i>old_itobj</i>	74
parser_listo (Var)	69	Oler (Acc): <i>Smell</i>	45
PreguntaCualExactamente (Var)	74	OOPS1__WD (Cons): <i>OOPS1__WD</i>	105
PreguntarPreposicion (Rut)	89	OOPS2__WD (Cons): <i>OOPS2__WD</i>	105
PreguntaSiNo (Var)	70	OOPS3__WD (Cons): <i>OOPS3__WD</i>	105
quitacentos (Var)	75	Orden (Acc): <i>Order</i>	64
QuitandoRFinal (Rut)	93	OrdenarJuntos (Rut): <i>SortTogether</i>	93
QuitarAcentos (Rut)	93	OrdenarLista (Rut): <i>SortOutList</i>	93
Salidas (Acc)	60	ordenes (Prop): <i>orders</i>	28
salidas (Prop)	32	OSCURIDAD__TX (Cons): <i>DARKNESS__TX</i>	105
Subir (Acc)	61	OTRAVEZ1__WD (Cons): <i>AGAIN1__WD</i> ...	105
NULL (Cons): <i>NULL</i>	98	OTRAVEZ2__WD (Cons): <i>AGAIN2__WD</i> ...	105
Num__DirDicc (Rut): <i>No__Dword</i>	93	OTRAVEZ3__WD (Cons): <i>AGAIN3__WD</i> ...	105
num_palabras (Var): <i>num_words</i>	73	otro (Var): <i>second</i>	69
numero_de_clases (Var): <i>number_of_classes</i> ...	73	OTRO1__WD (Cons): <i>OTHER1__WD</i>	105
numero_de_encajados (Var): <i>number_matched</i> ..	74	OTRO2__WD (Cons): <i>OTHER2__WD</i>	105
numero_especial (Var): <i>special_number</i>	74	OTRO3__WD (Cons): <i>OTHER3__WD</i>	105
numero_especial1 (Var): <i>special_number1</i>	74	OTRO_BIT (Cons): <i>OTHER_BIT</i>	109
numero_especial2 (Var): <i>special_number2</i>	74		

reaccionar_despues (Prop): <i>react_after</i>	32	Si (Acc): <i>Yes</i>	47
Recibir (Acc): <i>Receive</i>	65	SI1__WD (Cons): <i>YES1__WD</i>	106
RecibirLanzamiento (Acc): <i>ThrownAt</i>	65	SI2__WD (Cons): <i>YES2__WD</i>	107
recipiente (Atr): <i>container</i>	15	SI3__WD (Cons): <i>YES3__WD</i>	107
RECUPERAR__WD (Cons): <i>RESTORE__WD</i>	106	si_abierto (Prop): <i>when_open</i>	32
RECURSIVO_BIT (Cons): <i>RECURSE_BIT</i>	100	si_apagado (Prop): <i>when_off</i>	33
regla_coger_todo (Var): <i>take_all_rule</i>	75	si_cerrado (Prop): <i>when_closed</i>	33
Reiniciar (Acc): <i>Restart</i>	40	si_encendido (Prop): <i>when_on</i>	34
REINICIAR__WD (Cons): <i>RESTART__WD</i>	106	sibling (Rut): <i>sibling</i>	81
ReiniciarPalabrasVagas (Rut): <i>ResetVagueWords</i>	93	SIEMPRE_BIT (Cons): <i>SIEMPRE_BIT</i>	100
Release (Cons): <i>Release</i>	97	SiguienteElemento (Rut): <i>NextEntry</i>	94
relojes_activos (Var): <i>active_timers</i>	75	SiguientePalabra (Rut): <i>NextWord</i>	86
Responder (Acc): <i>Answer</i>	46	SiguientePalabraParar (Rut): <i>NextWordStopped</i>	86
Restaurar (Acc): <i>Restore</i>	41	SIN_DIRECCIONES (Cons): <i>WITHOUT_DIRECTIONS</i>	97
Retorcer (Acc): <i>Squeeze</i>	47	SINARTICULO_BIT (Cons): <i>NOARTICLE_BIT</i>	100
RevisarMulti (Rut): <i>ReviseMulti</i>	93	SiONo (Rut): <i>YesOrNo</i>	86
RevisionLib (Cons): <i>LibRelease</i>	109	Soplar (Acc): <i>Blow</i>	47
Rezar (Acc): <i>Pray</i>	47	soporte (Atr): <i>supporter</i>	15
RKEY__TX (Cons): <i>RKEY__TX</i>	106	Soso (Acc): <i>Mild</i>	47
RPG_CRIATURA (Cons): <i>GPR_CREATURE</i>	109	STRICT_MODE (Cons): <i>STRICT_MODE</i>	97
RPG_FALLO (Cons): <i>GPR_FAIL</i>	109	Subir (Acc): <i>(Nuevo)</i>	61
RPG_MULTI (Cons): <i>GPR_MULTI</i>	110	suma_al_alcance (Prop): <i>add_to_scope</i>	34
RPG_MULTIDENTRO (Cons): <i>GPR_MULTIINSIDE</i>	110	SumarAlAlcance (Rut): <i>AddToScope</i>	86
RPG_MULTIEXCEPTO (Cons): <i>GPR_MULTIEXCEPTO</i>	110	SustraerMulti (Rut): <i>MultiSub</i>	94
RPG_MULTIOSEIDO (Cons): <i>GPR_MULTIOHELD</i>	110	T	
RPG_NOMBRE (Cons): <i>GPR_NOUN</i>	110	Tacos (Acc): <i>Strong</i>	47
RPG_NUMERO (Cons): <i>GPR_NUMBER</i>	110	TAMANIO_LISTA_ENCAJAN (Cons): <i>MATCH_LIST_SIZE</i>	110
RPG_POSEIDO (Cons): <i>GPR_HELD</i>	110	tamano_listando (Var): <i>listing_size</i>	75
RPG_PREPOSICION (Cons): <i>GPR_PREPOSITION</i>	110	tate_callao (Var): <i>keep_silent</i>	70
RPG_REPARSE (Cons): <i>GPR_REPARSE</i>	110	tdatos_encontrado (Var): <i>found_tdata</i>	75
RutinaMirar (Rut): <i>LookRoutine</i>	89	techo_de_visibilidad (Var): <i>visibility_ceiling</i>	75
RutinaPostJuego (Rut): <i>GamePostRoutine</i>	89	TECLAANT1__KY (Cons): <i>PKEY1__KY</i>	107
RutinaPreJuego (Rut): <i>GamePreRoutine</i>	89	TECLAANT2__KY (Cons): <i>PKEY2__KY</i>	107
RutinasAntes (Rut): <i>BeforeRoutines</i>	93	TECLAANT__TX (Cons): <i>PKEY__TX</i>	107
RutinasDespues (Rut): <i>AfterRoutines</i>	86, 94	Teclado (Rut): <i>Keyboard</i>	86
S		TECLASIG1__KY (Cons): <i>NKEY1__KY</i>	107
Sacar (Acc): <i>Remove</i>	60	TECLASIG2__KY (Cons): <i>NKEY2__KY</i>	107
Salidas (Acc): <i>(Nuevo)</i>	60	TECLASIG__TX (Cons): <i>NKEY__TX</i>	107
salidas (Prop): <i>(Nuevo)</i>	32	TERMINAR1__WD (Cons): <i>QUIT1__WD</i>	107
Salir (Acc): <i>Exit</i>	61	TERMINAR2__WD (Cons): <i>QUIT2__WD</i>	107
Salirse (Acc): <i>GetOff</i>	61	TestCriatura (Rut): <i>CreatureTest</i>	94
Saltar (Acc): <i>Jump</i>	47	tiempo_agotado (Prop): <i>time_out</i>	34
SaltarSobre (Acc): <i>JumpOver</i>	47	tiempo_restante (Prop): <i>time_left</i>	35
Salvar (Acc): <i>Save</i>	41	TieneFuenteDeLuz (Rut): <i>HasLightSource</i>	87
SALVO1__WD (Cons): <i>BUT1__WD</i>	106	tipoerror (Var): <i>etype</i>	75
SALVO2__WD (Cons): <i>BUT2__WD</i>	106	Tirar (Acc): <i>Pull</i>	48
SALVO3__WD (Cons): <i>BUT2__WD</i>	106	Titular (Cons): <i>Headline</i>	98
SeRefiere (Rut): <i>Refers</i>	94	Tocar (Acc): <i>Touch</i>	48
Serial (Cons): <i>Serial</i>	97	TODO1__WD (Cons): <i>ALL1__WD</i>	107
SeVeATraves (Rut): <i>IsSeeThrough</i>	94	TODO2__WD (Cons): <i>ALL2__WD</i>	107
		TODO3__WD (Cons): <i>ALL3__WD</i>	107

TODO4__WD (Cons): <i>ALL4__WD</i>	107	U	UnaDireccion (Rut): <i>ADirection</i>	94
TODO5__WD (Cons): <i>ALL5__WD</i>	107		uno (Var): <i>noun</i>	70
token_alcance (Var): <i>scope_token</i>	75			
TOKEN_CRIATURA (Cons): <i>CREATURE_TOKEN</i>	110	V		
TOKEN_ESPECIAL (Cons): <i>SPECIAL_TOKEN</i>	110		Vaciar (Acc): <i>Empty</i>	62
TOKEN_FINAL (Cons): <i>ENDIT_TOKEN</i>	110		VaciarEn (Acc): <i>EmptyT</i>	62
TOKEN_MULTI (Cons): <i>MULTI_TOKEN</i>	110		valepuntos (Atr): <i>scored</i>	15
TOKEN_MULTIDENTRO (Cons): <i>MULTIINSIDE_TOKEN</i>	110		valor_lj (Var): <i>lt_value</i>	75
TOKEN_MULTIEXCEPTO (Cons): <i>MULTIEXCEPT_TOKEN</i>	110		ValorDelPronombre (Rut): <i>PronounValue</i>	87
TOKEN_MULTIPOSEIDO (Cons): <i>MULTIHELD_TOKEN</i>	110		ValorOEjecutar (Rut): <i>ValueOrRun</i>	87
TOKEN_NOMBRE (Cons): <i>NOUN_TOKEN</i> ..	110		VERBO_PE (Cons): <i>VERB_PE</i>	102
TOKEN_NUMERO (Cons): <i>NUMBER_TOKEN</i>	110		VerboDesconocido (Rut): <i>UnknownVerb</i>	90
TOKEN_POSEIDO (Cons): <i>HELD_TOKEN</i> ..	110		Verificar (Acc): <i>Verify</i>	41
TOKEN_TEMA (Cons): <i>TOPIC_TOKEN</i>	111		VersionIdioma (Cons): <i>LanguageVersion</i>	111
Tokenise__ (Rut): <i>Tokenise__</i>	94	X	Vestir (Acc): <i>Wear</i>	62
TopeAlcanzable (Rut): <i>ScopeCeiling</i>	94		vida (Prop): <i>life</i>	35
Transferir (Acc): <i>Transfer</i>	61		visitado (Atr): <i>visited</i>	16
transparente (Atr): <i>transparent</i>	15			
TrasElPrompt (Rut): <i>AfterPrompt</i>	89		x_cuenta_ambito (Var): <i>x_scope_count</i>	75
TrazarAccion (Rut): <i>TraceAction</i>	94		XArbol (Acc): <i>XTree</i>	63
Tregar (Acc): <i>Climb</i>	48		xcommsdir (Var): <i>xcommsdir</i>	75
true (Cons): <i>true</i>	98		XCompruebaMover (Rut): <i>XTestMove</i>	94
TT_ALCANCE (Cons): <i>SCOPE_TT</i>	111		XIrA (Acc): <i>Goto</i>	63
TT_ELEMENTAL (Cons): <i>ELEMENTARY_TT</i>	111		XMover (Acc): <i>XAbstract</i>	63
TT_FILTRO_ATRIB (Cons): <i>ATTR_FILTER_TT</i>	111	Y	XObj (Rut): <i>XObj</i>	94
TT_FILTRO Rutina (Cons): <i>ROUTINE_FILTER_TT</i>	111		XRobar (Acc): <i>XPurloin</i>	63
TT_ILEGAL (Cons): <i>ILLEGAL_TT</i>	111			
TT_PREPOSICION (Cons): <i>PREPOSITION_TT</i>	111		Y1__WD (Cons): <i>AND1__WD</i>	107
TT_RPG (Cons): <i>GPR_TT</i>	111		Y2__WD (Cons): <i>AND2__WD</i>	107
ttipo_encontrado (Var): <i>found_ttype</i>	75		Y3__WD (Cons): <i>AND3__WD</i>	107
TUMISMO__TX (Cons): <i>YOURSELF__TX</i> ..	107		Y__TX (Cons): <i>AND__TX</i>	108
turnos (Var): <i>turns</i>	70		YANOPRON_PE (Cons): <i>IT_GONE</i>	102
			YO1__WD (Cons): <i>ME1__WD</i>	108
			YO2__WD (Cons): <i>ME2__WD</i>	108
			YO3__WD (Cons): <i>ME3__WD</i>	108
		Z		
			ZRegion (Rut): <i>ZRegion</i>	87

10.4. Verbos de la gramática

A continuación se muestra una lista de todos los verbos que el jugador, por defecto, puede intentar en el juego. A pesar de que son muchísimos, gran parte de ellos son equivalentes entre sí, pues dan origen a la misma acción. Los números de página que figuran a continuación de cada verbo se refieren a la página donde se describe la acción que se generará con ese verbo. Observar que con algunos verbos la acción generada será distinta según el tipo de objeto directo usado (por ejemplo, el verbo “X” a secas produce la acción *Salidas*, mientras que el verbo “X objeto” produce la acción *Examinar*).

A

ABAJO	53	BAJA POR < x >	56
ABANDONAR	39	BAJATE	61
ABRAZA < x >	41	BAJATE DE < x >	61
ABRAZA A < x >	42	BAJATE POR < x >	56
ABRE < x >	48	BALANCEATE EN < x >	42
ABRE A < x >	48	BEBE < x >	41
ABRE < x > CON < y >	59	BESA < x >	41
ABRILLANTA < x >	43	BESA A < x >	41
ABURRIDO	47	BOBO	47
ACABAR	39	BREVE	40
ADQUIERE < x >	42	BRINCA < x >	47
AFUERA	61	BRINCA	47
AFUERA DE < x >	61	BRINCA A < x >	56
AGITA < x >	41	BRINCA POR ENCIMA DE < x >	47
AGITA LA MANO	43	BRINCA SOBRE < x >	47
AGITA LAS MANOS	43	BUSCA < x > EN < y >	42
AJUSTA < x >	43	BUSCA EN < x > < y >	42
AJUSTA < x > A < y >	46	BUSCA EN < x >	49
AJUSTA < x > EN < y >	46	BUSCA EN < x > ACERCA DE < y >	42
ANDA	44	BUSCA EN < x > SOBRE < y >	42
ANDA HACIA < x >	53		
APAGA < x >	48		
APAGA A < x >	48		
APLASTA < x >	41		
APRIETA < x >	47		
APRIETA A < x >	47		
ARRIBA	53		
ARROJA < x >	44		
ARROJA < x > CONTRA < y >	44		
ASESINA < x >	41		
ATA < x >	41		
ATA < x > A < y >	41		
ATA A < x >	41		
ATA A < x > A < y >	41		
ATACA < x >	41		
ATACA A < x >	41		
ATORNILLA < x >	43		
ATRAVIESA < x >	56		

B

B	53		
BAJA DE < x >	61		

C

CAMBIA < x > A < y >	61
CAMINA	44
CAMINA HACIA < x >	53
CANTA	42
CARGAR	41
CAVA < x >	43
CAVA < x > CON < y >	43
CAVA EN < x >	43
CAVA EN < x > CON < y >	43
CHUPA < x >	46
CIERRA < x >	49
CIERRA < x > CON < y >	51
CIERRA < x > CON PESTILLO	51
COGE < x >	49
COGE A < x >	49
COGE < x > DE < y >	60
COLOCA CERROJO A < x >	51
COLOCA < x > DENTRO DE < y >	55
COLOCA < x > EN < y >	55, 58
COLOCA < x > ENCIMA DE < y >	58
COLOCA < x > SOBRE < y >	58

COLUMPIATE EN $\langle x \rangle$	42	DESTAPA $\langle x \rangle$	48
COME $\langle x \rangle$	50	DESTRUYE $\langle x \rangle$	41
COMETE $\langle x \rangle$	50	DI $\langle x \rangle$ A $\langle y \rangle$	46
COMPRA $\langle x \rangle$	42	DI A $\langle x \rangle$ $\langle y \rangle$	46
CONECTA $\langle x \rangle$	41, 52	DIALECTO	39
CONECTA $\langle x \rangle$ A $\langle y \rangle$	41	DIALECTO CASTELLANO	39
CONECTA $\langle x \rangle$ CON $\langle y \rangle$	41	DIALECTO SUDAMERICANO	39
CONSULTA A $\langle x \rangle$ SOBRE $\langle y \rangle$	46	DILE $\langle x \rangle$ A $\langle y \rangle$	46
CONSULTA $\langle x \rangle$ ACERCA DE $\langle y \rangle$	42	DILE A $\langle x \rangle$ $\langle y \rangle$	46
CONSULTA $\langle x \rangle$ EN $\langle y \rangle$	42	DISCULPA	45
CONSULTA $\langle x \rangle$ SOBRE $\langle y \rangle$	42	DUERME	42
CONSULTA $\langle x \rangle$ SOBRE $\langle y \rangle$	46		
CONSULTA SOBRE $\langle x \rangle$ A $\langle y \rangle$	46	E	53
CORRE	44	ECHA CERROJO A $\langle x \rangle$	51
CORRE HACIA $\langle x \rangle$	53	ECHA $\langle x \rangle$ EN $\langle y \rangle$	55, 58
CORTA $\langle x \rangle$	42	ECHATE EN $\langle x \rangle$	56
CORTA $\langle x \rangle$ CON $\langle y \rangle$	42	EMPUJA $\langle x \rangle$	42
CORTO	40	EMPUJA A $\langle x \rangle$	42
CRUZA $\langle x \rangle$	56	EMPUJA $\langle x \rangle$ HACIA $\langle y \rangle$	43
CRUZA	52	ENCHUFA $\langle x \rangle$ A $\langle y \rangle$	41
CRUZA A $\langle x \rangle$	56	ENCIENDE $\langle x \rangle$	46
CRUZA POR $\langle x \rangle$	56	ENCIENDE $\langle x \rangle$	52
CUBRE $\langle x \rangle$	49	ENLAZA $\langle x \rangle$ A $\langle y \rangle$	41
CUENTA $\langle x \rangle$ A $\langle y \rangle$	44	ENSEÑA $\langle x \rangle$ A $\langle y \rangle$	58
CUENTA SOBRE $\langle x \rangle$ A $\langle y \rangle$	44	ENSEÑA A $\langle x \rangle$ $\langle y \rangle$	58
D		ENTRA $\langle x \rangle$	56
DA $\langle x \rangle$ A $\langle y \rangle$	50	ENTRA	52
DA A $\langle x \rangle$ $\langle y \rangle$	50	ENTRA A $\langle x \rangle$	56
DA UN GOLPE A $\langle x \rangle$	41	ENTRA EN $\langle x \rangle$	56
DA UN PUÑETAZO A $\langle x \rangle$	41	ENTRA POR $\langle x \rangle$	56
DA UNA PATADA A $\langle x \rangle$	41	ESCALA $\langle x \rangle$	48
DALE $\langle x \rangle$ A $\langle y \rangle$	50	ESCALA A $\langle x \rangle$	48
DALE A $\langle x \rangle$ $\langle y \rangle$	50	ESCALA POR $\langle x \rangle$	48
DASELA A $\langle x \rangle$	50	ESCUCHA $\langle x \rangle$	43
DASELO A $\langle x \rangle$	50	ESCUCHA	43
DEJA $\langle x \rangle$	51	ESCUCHA A $\langle x \rangle$	43
DEJA A $\langle x \rangle$	51	ESPABILA	42
DEJA A $\langle x \rangle$ EN $\langle y \rangle$	58	ESPABILA A $\langle x \rangle$	42
DEJA $\langle x \rangle$ DENTRO DE $\langle y \rangle$	55	ESPABILATE	42
DEJA $\langle x \rangle$ EN $\langle y \rangle$	58	ESPERA	43
DEJA $\langle x \rangle$ EN $\langle y \rangle$	55	ESTE	53
DESATORNILLA $\langle x \rangle$	43	ESTRUJA $\langle x \rangle$	47
DESCANSA	42	EX $\langle x \rangle$	52
DESCONECTA $\langle x \rangle$	48	EXAMINA $\langle x \rangle$	52
DESCONECTA A $\langle x \rangle$	48	EXAMINA A $\langle x \rangle$	52
DESCRIBE $\langle x \rangle$	52	EXCAVA $\langle x \rangle$	43
DESCUBRE $\langle x \rangle$	48	EXCAVA $\langle x \rangle$ CON $\langle y \rangle$	43
DESPIERTA $\langle x \rangle$	42	EXCAVA EN $\langle x \rangle$	43
DESPIERTA	42	EXCAVA EN $\langle x \rangle$ CON $\langle y \rangle$	43
DESPIERTA A $\langle x \rangle$	42	EXITS	60
DESPIERTATE	42	EXPLICA $\langle x \rangle$ A $\langle y \rangle$	44
DESPLAZA $\langle x \rangle$	42	EXPLICA A $\langle x \rangle$ $\langle y \rangle$	44
DESPLAZA A $\langle x \rangle$	42		
DESPLAZA $\langle x \rangle$ HACIA $\langle y \rangle$	43		

F

FIJA < x >	43
FIJA < x > A < y >	46
FIJA < x > EN < y >	46
FIN	39
FRIEGA < x >	43
FROTA < x >	43
FUERA	61
FUERA DE < x >	61

G

GIRA < x >	43
GOLPEA < x >	41
GUARDAR	41

H

HABLA < x > < y >	44
HABLA < x > A < y >	44
HABLA A < x > DE < y >	44
HABLA A < x > SOBRE < y >	44
HABLA CON < x > ACERCA DE < y >	44
HABLA CON < x > DE < y >	44
HABLA CON < x > SOBRE < y >	44
HABLA < x > DE < y >	44
HABLA DE < x > A < y >	44
HABLA DE < x > CON < y >	44
HABLA < x > SOBRE < y >	44
HABLA SOBRE < x > CON < y >	44
HUELE < x >	45
HUELE	45
HUELE A < x >	45

I

I	53
I ANCHO	53
I BREVE	53
I ESTRECHO	53
I LARGO	53
IDIOTA	47
INGIERE < x >	50
INSERTA < x > EN < y >	55, 58
INSPECCIONA < x >	52
INV	53
INV ANCHO	53
INV BREVE	53
INV ESTRECHO	53
INV LARGO	53
INVENTARIO	53
INVENTARIO ANCHO	53
INVENTARIO BREVE	53
INVENTARIO ESTRECHO	53
INVENTARIO LARGO	53

J

JODER	47
-------------	----

L

L	57
LAME < x >	46
LAME A < x >	46
LAMENTO	45
LANZA < x > A < y >	44
LANZA A < x > CONTRA < y >	44
LANZA A < x > POR < y >	44
LANZA < x > CONTRA < y >	44
LANZA < x > POR < y >	44
LARGO	40
LAVA < x >	43
LAVA A < x >	43
LEE < x >	52
LEE < x > EN < y >	42
LEE SOBRE < x > EN < y >	42
LEVANTATE	61
LEVANTATE DE < x >	61
LIMPIA < x >	43
LLENA < x >	44
LO LAMENTO	45
LO SIENTO	45
LOAD	39, 41
LOOK	57
LUCHA CON < x >	41
LUGARES	40

M

M	56
MASTICA < x >	50
MATA < x >	41
MATA A < x >	41
MENEA < x >	42
MENEA A < x >	42
MENEATE EN < x >	42
METE < x > DENTRO DE < y >	55
METE < x > EN < y >	55, 58
METETE EN < x >	56
METETE POR < x >	56
MIERDA	47
MIRA < x >	52
MIRA A < x >	52
MIRA A TRAVES DE < x >	49
MIRA BAJO < x >	45
MIRA DEBAJO DE < x >	45
MIRA DENTRO DE < x >	49
MIRA EN < x >	49
MIRA HACIA < x >	52
MIRA POR < x >	49
MIRA SOBRE < x >	49
MIRAR	56
MUESTRA < x > < y >	58
MUESTRA < x > A < y >	58
MUESTRA A < x > < y >	58

MUEVE < x >	42	PIDELE A < x > < y >	45
MUEVE A < x >	42	PIENSA	46
MUEVE < x > HACIA < y >	43	PISOTEA < x >	41
N		PLACES	40
N	53	PON < x > A < y >	46
NADA	45	PON A < x > DENTRO DE < y >	55
NARRA < x > A < y >	44	PON A < x > EN < y >	55
NARRA A < x > < y >	44	PON A < x > EN < y >	58
NO	45, 53	PON A < x > ENCIMA DE < y >	58
NOQUEA < x >	41	PON A < x > SOBRE < y >	58
NORDESTE	53	PON CERROJO A < x >	51
NORESTE	53	PON < x > DENTRO DE < y >	55
NORMAL	40	PON < x > EN < y >	58
NOROESTE	53	PON < x > EN < y >	55
NORTE	53	PON < x > EN < y >	55
NOSCRIPT	39	PON < x > ENCIMA DE < y >	58
NOTIFICAR NO	39	PON < x > SOBRE < y >	58
NOTIFICAR OFF	39	PONTE < x >	62
NOTIFICAR ON	39	PREGUNTA < x > A < y >	46
NOTIFICAR SI	39	PREGUNTA A < x > ACERCA DE < y >	46
NOTIFY NO	39	PREGUNTA A < x > POR < y >	46
NOTIFY OFF	39	PREGUNTA A < x > SOBRE < y >	46
NOTIFY ON	39	PREGUNTA < x > POR < y >	46
NOTIFY SI	39	PREGUNTA POR < x > A < y >	46
NOTRANSCRIPCION	39	PREGUNTA < x > SOBRE < y >	46
NW	53	PREGUNTA SOBRE < x > A < y >	46
NX	45	PRENDE < x >	46
O		PRENDE < x >	52
O	53	PRONOMBRES	40
OBJECTS	40	PRUEBA < x >	46
OBJETOS	40	PT	40
OBSERVA < x >	52	PULE < x >	43
OFRECE < x > A < y >	50	PULSA < x >	43
OFRECE A < x > < y >	50	PUNTOS	40
OLFATEA < x >	45	PUNTUACION	40
OLFATEA	45	PUNTUACION DETALLADA	40
OLFATEA A < x >	45	PUNTUACION TOTAL	40
ONDEA < x >	41	Q	
ORA	47	Q	39
OYE < x >	43	QUEMA < x >	46
OYE	43	QUEMA A < x >	46
P		QUEMA A < x > CON < y >	46
PALADEA < x >	46	QUEMA < x > CON < y >	46
PALPA < x >	48	QUIT	39
PALPA A < x >	48	QUITA < x > A < y >	49
PASA POR < x >	56	QUITA A < x > DE < y >	60
PATEA < x >	41	QUITA CERROJO A < x >	59
PERDON	45	QUITA CIERRE A < x >	59
PERDONA	45	QUITA < x > DE < y >	60
PIDE < x > A < y >	45	QUITA PESTILLO A < x >	59
PIDE A < x > < y >	45	QUITALE EL CERROJO A < x >	59
PIDELE < x > A < y >	45	QUITALE EL CIERRE A < x >	59
		QUITALE EL PESTILLO A < x >	59
		QUITATE < x >	51

R

R	53	SI	45, 47
RASGA < x >	42	SIENTA EN < x >	56
RASGA < x > CON < y >	42	SIENTATE EN < x >	56
REBUSCA EN < x >	49	SIENTO	45
RECUPERAR	41	SO	53
REGALA < x > A < y >	50	SOPLA < x >	47
REGALA A < x > < y >	50	SORRY	45
REGISTRA < x >	49	SUBE	61
REGISTRA A < x >	49	SUBE A < x >	56
REGISTRA EN < x >	49	SUBE EN < x >	56
REINICIAR	40	SUBE POR < x >	56
RELLENA < x >	44	SUBETE A < x >	56
RESPONDE < x > A < y >	46	SUBETE EN < x >	56
RESPONDE A < x > < y >	46	SUDESTE	53
RESTAURAR	41	SUELTA < x > DENTRO DE < y >	55
RESTORE	41	SUELTA < x > EN < y >	55
RETUERCE < x >	47	SUELTA < x > ENCIMA DE < y >	58
REZA	47	SUELTA < x > SOBRE < y >	58
ROMPE < x >	41	SUPERBREVE	40
RONCA	42	SUR	53
		SURESTE	53
		SUROESTE	53
		SW < y >	53

S

S	53
SABOREA < x >	46
SACA A < x > DE < y >	60
SACA < x > DE < y >	60
SACATE < x >	51
SACUDE < x >	41
SACUDE < x >	41
SACUDE LA MANO	43
SACUDE LAS MANOS	43
SAL	61
SAL AFUERA	61
SAL DE < x >	61
SAL FUERA	61
SAL POR < x >	56
SALIDAS	60
SALTA < x >	47
SALTA	47
SALTA A < x >	56
SALTA POR ENCIMA DE < x >	47
SALTA SOBRE < x >	47
SALTE	61
SALTE DE < x >	61
SALTE POR < x >	56
SALUDA CON LA MANO	43
SALVAR	41
SAVE	39, 41
SCRIPT	39
SCRIPT NO	39
SCRIPT OFF	39
SCRIPT ON	39
SCRIPT SI	39
SE	53

T

TAPA < x >	49
TERMINAR	39
TIRA < x > . < y >	51
TIRA < x > A < y >	55
TIRA < x > A < y >	44
TIRA < x > CONTRA < y >	44
TIRA DE < x >	48
TIRA < x > DENTRO DE < y >	55
TIRA < x > EN < y >	58
TIRA < x > EN < y >	55
TIRA < x > ENCIMA DE < y >	58
TIRA < x > POR < y >	55
TIRA < x > SOBRE < y >	58
TOCA < x >	48
TOCA A < x >	48
TOMA < x >	49
TOMA A < x >	49
TOMA < x > DE < y >	60
TONTO	47
TORTURA < x >	41
TOTAL	40
TRAGA < x >	50
TRANSCRIPCION	39
TRANSCRIPCION NO	39
TRANSCRIPCION OFF	39
TRANSCRIPCION ON	39
TRANSCRIPCION SI	39
TRANSFIERE < x > A < y >	61
TREPA < x >	48
TREPA A < x >	48

TREPA POR $\langle x \rangle$	48	VETE	44
TUERCE $\langle x \rangle$	47	VETE A $\langle x \rangle$	56
U		VETE A $\langle x \rangle$	53
UNE $\langle x \rangle$ A $\langle y \rangle$	41	VETE HACIA $\langle x \rangle$	56
UNSCRIPT	39	VETE HACIA $\langle x \rangle$	53
V		VETE POR $\langle x \rangle$	56
VACIA $\langle x \rangle$	62	VISTE $\langle x \rangle$	62
VACIA $\langle x \rangle$ DENTRO DE $\langle y \rangle$	62	VISTETE CON $\langle x \rangle$. $\langle y \rangle$	62
VACIA $\langle x \rangle$ EN $\langle y \rangle$	62	VUELVE	44
VACIA $\langle x \rangle$ ENCIMA DE $\langle y \rangle$	62	VUELVE HACIA $\langle x \rangle$	53
VACIA $\langle x \rangle$ SOBRE $\langle y \rangle$	62	W	
VE $\langle x \rangle$	53	W	53
VE	44	X	
VE A $\langle x \rangle$	53	X $\langle x \rangle$	52
VE HACIA $\langle x \rangle$	53	X	60
VERBOSE	40	Z	
VERIFICAR	41	Z	43
VETE $\langle x \rangle$	56		
VETE $\langle x \rangle$	53		