Esquema general de Minë
Una guía para el colaborador

versión 1.0

5 de Noviembre de 2002

Siempre actualizada en http://eutherpe.org

Pablo Ruiz Múzquiz *Aranarth*,
Andrés Moya Velázquez *Hirunatan*
y Diego Moya Velázquez *Hizgael*

Ancho, alto y profundo es el reino de los cuentos de hadas, y lleno todo él de cosas diversas: hay allí toda suerte de bestias y pájaros; mares sin riberas e incontables estrellas; belleza que embelesa y un peligro siempre presente; la alegría, lo mismo que la tristeza, son afiladas como espadas. Tal vez un hombre pueda sentirse dichoso de haber vagado por ese reino, pero su misma plenitud y condición arcana atan la lengua del viajero que desee describirlo. Y mientras está en él le resulta peligroso hacer demasiadas preguntas, no vaya a ser que las puertas se cierren y desaparezcan las llaves.

Sobre los Cuentos de Hadas, JRR Tolkien

Índice general

Histórico de cambios

5 noviembre 2002:

- Añadido histórico de cambios.
- 4.2.3 (Las salidas en una sala): añadida llave en tipo_cierre=”horario”
- 4.2.1 (Datos generales de toda sala): Incrementada versión del sistema de salas a 1.2.

1 Introducción

1 Introducción

En esta *guía* de Minë pretendemos elaborar un documento que facilite al máximo la contribución de código y de ficheros de descripción de salas, objetos y personajes no jugadores (PNJs).

Para ello, hemos dividido la guía en dos apartados muy diferenciados; en el primero hablamos de cómo puede construirse un área de juego completa con salas y objetos pero sin PNJs y en el segundo describimos el diseño del programa a nivel de código fuente para dar a conocer las entrañas y animar a los programadores a participar en el desarrollo del núcleo. Además, acompañamos todo esto con varios apéndices que sirven de apoyo a estas dos primeras partes.

Los conocimientos o el trabajo derivado del uso de esta guía no ha de repercutir necesariamente en Minë. Ni el equipo de Minë tomará por la fuerza nada que una persona ajena haya construido ni estamos obligados aceptar cualquier trabajo o estudio realizado en este sentido. La única restricción que imponemos es que las modificaciones realizadas a una copia privada del código fuente de Minë han de acogerse también a la Licencia Pública General (visite http://www.gnu.org).

Si no tienes conocimientos de programación, sólo has de leer la sección 2 y los apéndices 4.1, 4.2 y 4.3. Si deseas contribuir con código fuente, es necesario que leas por encima primero las partes dedicadas al núcleo en sí -sección 3 y apéndices 4.4 y 4.5-, luego leas las secciones dedicadas a los no programadores y luego vuelvas ya en serio a tus apartados.

Los autores de esta guía somos Andrés *Hirunatan*, Diego *Hizgael* y yo mismo, Pablo *Aranarth*. Cada uno se ha dedicado a un aspecto, Andrés a explicar el código fuente de Minë, Diego a todo lo relacionado con los personajes no jugadores (PNJs) y yo para los manuales, los comentarios y el armazón final del documento.

Quisiéramos agradecer al equipo que forma el proyecto Minë su apoyo mostrado hasta ahora[1]. Ellos son los suscritos a la lista de correo de usuarios.

Esperamos que de alguna forma u otra (jugando, informando de fallos, escribiendo áreas o código) disfrutéis de Minë tanto como nosotros.

[1] El logotipo de Minë es obra de Jaime 'Dwalin'.

2 Construcción de un área

2.1. Conceptos generales

Minë es un juego tipo MUD/MUSH. Esto quiere decir que el ordenador que ejecuta Minë construye un mundo virtual al que pueden conectarse otros jugadores por internet (o red local). La forma en que los jugadores ven el mundo de Minë es muy peculiar ya que, al contrario de lo acostumbrado, el entorno de juego no es gráfico sino textual.

De hecho, cuando un jugador se conecta al servidor que tiene Minë[1], lo primero que ve es un mensaje de bienvenida en texto simple. Este modelo de traspaso de información se mantiene siempre. El jugador escribe lo que desea hacer y Minë responde con texto. Para aquellos que frecuenten los *chats* (sobre todo IRC) resultará fácil hacerse una idea de lo que estamos diciendo.

En un **juego de rol**, los jugadores se desplazan por lugares muy diversos. Pueden estar andando por un camino, registrando una casa, inspeccionando una posada, vagando por un laberinto o escalando una montaña. En todos estos lugares el Director de Juego *divide* mentalmente el área de juego en pequeñas secciones. A veces no es obligatorio que el director de juego tenga diferenciados cada trozo del camino como si fuesen independientes y prefiere, todo lo más, tener clara la entrada al camino, un punto concreto de éste, el resto del camino y la salida o bifurcación más cercana. Si el Director de Juego fuese estricto debería informar a los jugadores de que según van avanzando, van pasando de un trozo del camino (llamémoslo parte 4 del camino) al siguiente (parte 5 del camino) pero esto raras veces sucede pues sacrifica demasiada jugabilidad. Si nos encontramos en una casa, la situación cambia radicalmente y solemos tener delante un pequeño mapa con las salas o cuartos bien diferenciados, ya que si sucede algo inesperado, no es lo mismo estar cerca de la escalera que al final del pasillo, es decir, cada lugar cuenta.

Cuando la situación la controla un ordenador en vez de una persona, todos los casos han de manejarse como la situación en la casa descrita arriba pues éste es incapaz de diferenciar cuándo es importante la división en zonas del terreno y cuándo no. Por tanto, en Minë hay que documentar cada trozo más o menos grande de terreno. Veamos, el objetivo es construir una Tierra Media virtual que contenga en su interior detalles de un cuarto trastero de una taberna de un poblacho del sur del extinto reino de Cardolan, pero también que haga mención explícita a un recodo en el camino que cruza el Anduin a la altura de Minas Tirith o un pequeño oasis muy al sur de Harad que sirve de descanso para los comerciantes de la zona. Nosotros nos centraremos en una pequeña casa anónima de un poblado del Reino de Arthedain alrededor del año 2000 de la Tercera Edad.

El propósito de esta guía es construir paso a paso cada *sala* del lugar, así como los objetos contenidos en ella. Para seguir correctamente el tutorial es muy importante que tengáis en todo momento una buena imagen mental del lugar que ocupamos en cada instante.

Las salas se componen básicamente de 4 cosas.

- Descripción
  - del lugar
  - detalles del lugar
- Objetos existentes
- Personajes no jugadores presentes
- Salidas posibles

[1] el servidor es una máquina física que ejecuta Minë y a la que se puede acceder de alguna forma con un programa específi co.

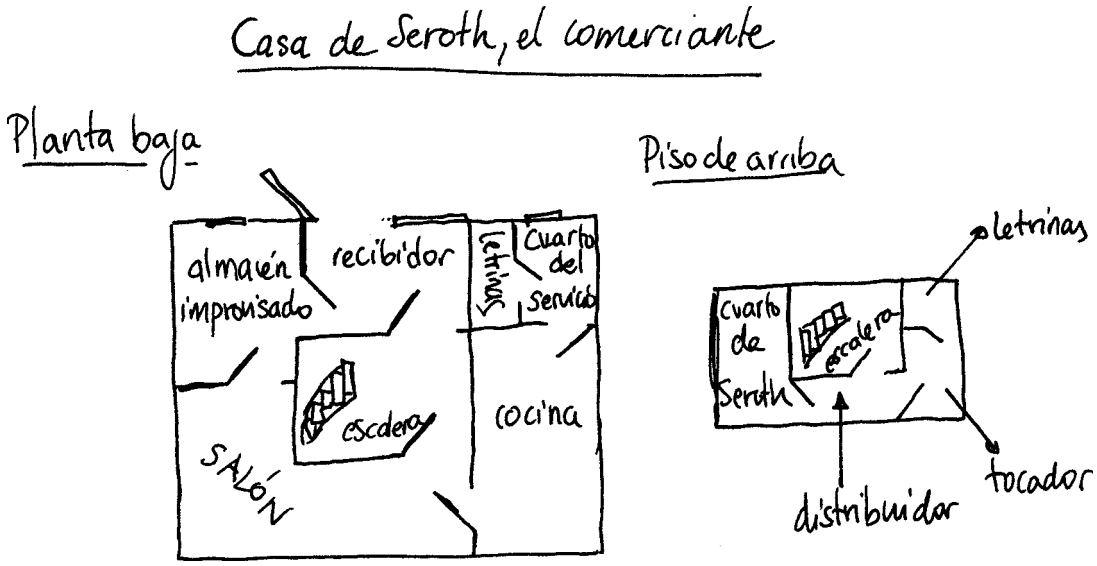
2.2. Cómo dar forma a la imaginación

Todo diseño de un área comienza con un dibujo a mano alzada de ésta para tener siempre claro la relación entre los cuartos o espacios considerados independientes. Es cierto que hay gente que posee una gran visión espacial y memoria pero para la mayoría es recomendable seguir en el papel todo lo que hacemos por escrito.

Como podéis ver en la figura 1 de una casa inventada que hemos llamado "Casa de Seroth, el comerciante" y a la que nos referiremos a partir de ahora como área **seroth**, nos hemos limitado a dibujar dos plantas, la de abajo -al nivel de la calle- y la de arriba -un simple y pequeño primer piso al que se accede por medio de una escalera de caracol-. En este ejemplo no pretendemos ser fieles a lo que sería una casa de un comerciante de telas en Arthedain y obviaremos muchos detalles.

Lo que hay en cada habitación o **sala** no aparece dibujado porque no interesa ni merece la pena. Como ya hemos dicho, lo importante es dominar la ubicación de los cuartos a fin de tener claro desde dónde se puede venir al lugar en que nos encontramos o hacia dónde podemos irnos desde una posición concreta. La descripción detallada se deja para la redacción posterior.

Por tanto, en el dibujo deben quedar claras la colocación de las salas y las salidas de que disponen, marcando aquellas que estén ocultas o sean difíciles de detectar así como aquellas que se encuentren cerradas con llave o con algún tipo de encantamiento (esto último resultará muy raro de encontrar).



2.3. Pasamos a la acción. ¿Cómo escribo una habitación?

Los detalles de cómo es el fichero de una sala se describen en el Apéndice titulado **Cómo construir una sala**.

En esta sección pondremos el contenido de cada fichero asociado a una sala.

Para escribir estos ficheros hemos utilizado un editor de textos sencillo, capaz de escribir en formato TXT simple y como guía el mencionado manual de los apéndices.

Creamos un directorio de nombre el del área que vayamos a crear. En nuestro caso, creamos un directorio llamado **seroth.** Dentro de este directorio vamos creando diferentes ficheros con diferentes nombres, siempre de acuerdo al manual mencionado. Veamos nuestro ejemplo.

2.3.1. Planta baja

Recibidor

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
<id>seroth-recibidor01</id>
<nombre>el recibidor de la casa de Seroth</nombre>

<descripcion>

<item>Se trata de un recibidor relativamente amplio pero sin muchos
adornos. A la izquierda hay una especie de cuadro que parace
representar el árbol genealógico del dueño de la casa y el suelo se
encuentra cubierto con una alfombra algo gastada por el continuo
trajín del envío de mercancías.</item>

<item>Al oeste divisas algo que parece ser un almacén bastante amplio
por la cantidad de cajas apiladas, y al sur justo hay un cuartito en
penumbra bastante pequeño. En el norte está la salida de la casa, que
da a la calle.</item>

</descripcion>

<salidas>
<sur id="seroth-escaleras01">un cuarto en penumbra</sur>
<oeste id="seroth-almacen01">un cuarto amplio que parece un
almacén.</oeste>
</salidas>

<propiedades tipo='edificio' subtipo='vestíbulo' volumen='6' luz='65'
   aura='55' combate="si" ocultabilidad='05'/>

<objetos></objetos>
<encuentros></encuentros>
</sala>
```

Almacén

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
<id>seroth-almacen01</id>
<nombre>el almacén de la casa de Seroth</nombre>

<descripcion>

<item>
Realmente, este lugar no fue diseñado para ser un almacén pero acabó
por destinarse a esta tarea a la vista de las innumerables cajas que,
apiladas, se amontan pegadas a las paredes. Seroth se cuida de que sus
frágiles productos no se dañen por malas manipulaciones con el
transporte y aquí todo parece estar bastante ordenado. Notas un olor
intenso a perfumes mezclados entre los que destaca, curiosamente, el
tomillo.
</item>

<item>Al sur hay una puerta cerrada y al este está la entrada</item>

<item dificultad="25">Uno de los cristales de la ventana que da al
norte está agrietado.</item> </descripcion>

<salidas>
<sur id="seroth-salon01">una puerta cerrada</sur>
<este id="seroth-recibidor01">el recibidor de Seroth</este>
</salidas>

<propiedades tipo="edificio" subtipo="habitación" volumen='13' luz='55'
  aura='50' combate="si" ocultabilidad='25'/>

<objetos>
<objeto id="tomillo01" cantidad="5" probabilidad="10">tomillo</objeto>

</objetos>
<encuentros></encuentros>
</sala>
```

Salón

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
<id>seroth-salon01</id>
<nombre>el salón de la casa de Seroth</nombre>

<descripcion>

<item>Sin duda, ésta debe ser la habitación más lujosa de la
casa. Aparte de dos enormes sofás apoyados contra las paredes sur y
oeste de la sala y que no tienen el aspecto de haber sido fabricados
por la región, hay una mesita que los separa con un par de libros de
```

```
tapa roja aterciopelada. En las paredes cuelgan media docena de
estupendos cuadros enmarcados todos ellos con madera de nogal
añeja. El suelo lo viste una alfombra de ricos detalles traída del
lejano sur y en el techo cuelga una enorme lámpara de cristal y bronce
con la mitad de las velas encendidas para asegurar la luz a esta zona
de la casa.</item>

<item>Tanto al norte como al oeste hay dos puertas cerradas... Al
noreste hay una puertecita abierta que da a unas escaleras de
caracol.</item>

<item dificultad="25">Los dos libros que hay encima de la mesita
tratan temas relacionados con la venta de pieles en Gondor.</item>

</descripcion>

<salidas>
<norte id="seroth-almacen01">una puerta cerrada</norte>
<este id="seroth-cocina01">una puerta cerrada</este>
<noreste id="seroth-escaleras01">unas escaleras de caracol</noreste>
</salidas>

<propiedades tipo='edificio' subtipo='salón' volumen='25' luz='45'
   aura='53' combate="si" ocultabilidad='25'/>

<objetos>
</objetos>
<encuentros></encuentros>
</sala>
```

Cocina

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
<id>seroth-cocina01</id>
<nombre>la cocina de la casa de Seroth</nombre>

<descripcion>

<item>
La cocina de Seroth es bastante limpia y contiene todo lo necesario
para que el servicio cocine muy diferentes platos. Hay un brasero, un
horno de barro, muchos trapos colgados de pequeños clavos. El suelo
está doblemente barnizado para evitar que la madera se estropee. En un
armarito cerrado con llave y con puertas translúcidas se pueden ver
dos cuberterías completas y una vajilla de excelente calidad. Existe
aquí un olor intenso a especias como el azafrán y tomillo (que inunda
toda la casa).
</item>

<item>
```

```
Al oeste hay una puerta cerrada y al norte una puerta que da al cuarto
del servicio de la casa.

</item>
</descripcion>

<salidas>
<norte id="seroth-servicio01">el cuarto del servicio</norte>
<oeste id="seroth-salon01">una puerta cerrada</oeste>
</salidas>

<propiedades tipo="edificio" subtipo="cocina" volumen='20' luz='55'
   aura='50' combate="si" ocultabilidad='18'/>

<objetos>
<objeto id="cuchillo01" cantidad="3" probabilidad="10">un cuchillo</objeto>
</objetos>
<encuentros></encuentros>
</sala>
```

Cuarto del servicio

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
<id>seroth-servicio01</id>
<nombre>el cuarto del servicio de la casa de Seroth</nombre>

<descripcion>

<item>Es un cuartucho pequeño y ligeramente incómodo. Al norte hay un
ventanuco que da a la calle y por el que se filtran ruidos y la luz. El
camastro que copa casi todo el espacio está todo lo arreglado que se
puede estar y el suelo desnudo no se encuentra demasiado sucio. Del
oeste llega un olor ligeramente desagradable ya que ahí es donde están
las letrinas del servicio. El techo es aquí más bajo debido a uno
armarios que cubren casi la mitad de la altura y que están destinados
a guardar manteles y ropas viejas, nada de valor.</item>

<item>Al oeste hay una puerta muy pequeña que da a las letrinas y al
sur está la cocina.</item>

</descripcion>

<salidas>
<sur id="seroth-cocina01">la cocina</sur>
<oeste id="seroth-servicio_letrinas01">las letrinas del servicio</oeste>
</salidas>

<propiedades tipo='edificio' subtipo='habitación' volumen='5' luz='40'
   aura='48' combate="si" ocultabilidad='15'/>
```

```
<objetos>
</objetos>
<encuentros></encuentros>
</sala>
```

Letrinas del servicio

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
<id>seroth-servicio_letrinas01</id>
<nombre>las letrinas del servicio de la casa de Seroth</nombre>

<descripcion>

<item>Un sucio cuarto que cumple su función. Oscuro y un poco
maloliente aunque con un buen desague.</item>

<item>Sólo hay una salida, al este, que da al cuarto del
servicio.</item>

</descripcion>

<salidas>
    <este id="seroth-servicio01">el cuarto del servicio</este>
</salidas>

<propiedades tipo='edificio' subtipo='calabozo/letrina' volumen='2' luz='20'
   aura='47' combate="si" ocultabilidad='10'/>

<objetos>
</objetos>
<encuentros></encuentros>
</sala>
```

Escaleras

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
<id>seroth-escaleras01</id>
<nombre>las escaleras entre plantas</nombre>

<descripcion>

<item>Un cuarto pequeño con unas escaleras de caracol que suban a la
planta de arriba.</item>

<item>Puedes ir arriba, ir al norte y aparecer en el recibidor o ir al
sur donde hay una puerta cerrada.</item>
```

```
</descripcion>

<salidas>
<arriba id="seroth-escaleras02">la planta de arriba</arriba>
<norte id="seroth-recibidor01">el recibidor</norte>
<sur id="seroth-salon01">una puerta cerrada</sur>
</salidas>

<propiedades tipo='edificio' subtipo='escaleras' volumen='2' luz='20'
   aura='45' combate="si" ocultabilidad='30'/>

<objetos>
</objetos>
<encuentros></encuentros>
</sala>
```

2.3.2. Planta de arriba

Escaleras

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
<id>seroth-escaleras02</id>
<nombre>las escaleras entre plantas</nombre>

<descripcion>

<item>Un cuarto pequeño con unas escaleras de caracol que bajan a la
planta de abajo.</item>

<item>Puedes ir abajo o ir al sur y aparecer en el
distribuidor.</item>

</descripcion>

<salidas>
<abajo id="seroth-escaleras01">la planta de abajo</abajo>
<sur id="seroth-distribuidor01">el distribuidor</sur>
</salidas>

<propiedades tipo='edificio' subtipo='escaleras' volumen='2' luz='25'
   aura='45' combate="si" ocultabilidad='20'/>

<objetos>
</objetos>
<encuentros></encuentros>
</sala>
```

Distribuidor

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
<id>seroth-distribuidor01</id>
<nombre>el distribuidor de la planta de arriba</nombre>

<descripcion>

<item>
Se trata de un pequeño pasillo-distribuidor sin ninguna
decoración y con algunas lamas de madera del suelo en mal estado que
crujen a tu paso.
</item>

</descripcion>

<salidas>
<este id="seroth-tocador01">una puerta cerrada</este>
<oeste id="seroth-cuarto_seroth01">una puerta cerrada</oeste>
</salidas>

<propiedades tipo=’edificio’ subtipo=’pasillo’ volumen=’4’ luz=’45’
   aura=’45’ combate="si" ocultabilidad=’10’/>

<objetos>
</objetos>
<encuentros></encuentros>
</sala>
```

Cuarto de Seroth

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
<id>seroth-cuarto_seroth01</id>
<nombre>el cuarto privado de Seroth</nombre>

<descripcion>

<item>
Este cuarto tiene una cama bastante amplia y un pequeño buró con hojas
y una pluma de ave con tintero. Hacia el oeste ves un armario
empotrado y al norte, junto a la cama, hay una mesita de noche muy
extravagante, de tonos plateados y ocres. El techo es aquí más bajo
que en la planta inferior y las vigas de madera son accesibles para un
humano de mediana estatura. En general se respira un ambiente muy
agradable.
</item>

<item dificultad="50">Hay unas inscripciones en la mesita de noche que
```

```
dicen ’Regalo de Aldandil’.
</item>

</descripcion>

<salidas>
<este id="seroth-distribuidor01">una puerta cerrada</este>
</salidas>

<propiedades tipo=’edificio’ subtipo=’habitación’ volumen=’2’ luz=’45’
   aura=’50’ combate="si" ocultabilidad=’10’/>

<objetos>
<objeto id="pluma_de_ave01" cantidad="2" probabilidad="0">una pluma de ave</objeto>
</objetos>
<encuentros></encuentros>
</sala>
```

Tocador

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
<id>seroth-tocador01</id>
<nombre>el tocador de la planta de arriba</nombre>

<descripcion>

<item>Un antiguo tocador que ya apenas se usa. Hay un gran espejo
ricamente decorado en los bordes y un soporte vacío.</item>

</descripcion>

<salidas>
<norte id="seroth-letrinas01">las letrinas</norte>
<oeste id="seroth-distribuidor01">una puerta cerrada</oeste>
</salidas>

<propiedades tipo=’edificio’ subtipo=’cuarto sin ventanas’ volumen=’4’
   luz=’55’ aura=’55’ combate="si" ocultabilidad=’10’/>

<objetos>
</objetos>
<encuentros></encuentros>
</sala>
```

Letrinas

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
<sala version="1.2" autor="Aranarth" area="seroth">
```

```
<id>seroth-letrinas01</id>
<nombre>las letrinas de la planta de arriba</nombre>

<descripcion>

<item>Unas letrinas muy limpias con un desagüe escondido. El cuarto es
pequeño pero no llega a agobiar.</item>

</descripcion>

<salidas>
<sur id="seroth-tocador01">el tocador</sur>
</salidas>

<propiedades tipo=’edificio’ subtipo=’calabozo/letrina’ volumen=’2’
   luz=’45’ aura=’50’ combate="si" ocultabilidad=’10’/>

<objetos>
</objetos>
<encuentros></encuentros>
</sala>
```

2.4. ¿Qué hay de los objetos?

Recordemos que en un entorno MUD/MUSH podemos pensar en 4 tipos de entes fundamentalmente:

- Personajes jugadores. Son personas físicas que se conectan por internet y realizan acciones conscientemente.
- Personajes no jugadores. Son programas de ordenador capaces de reaccionar a eventos y que se aparecen a los otros personajes jugadores como ellos.
- Objetos. Son elementos virtuales que simulan ser objetos físicos que pueden ser manipulados por PJs o PNJs.
- Salas. Son los espacios físicos virtuales por donde circulan personajes jugadores (PJs) personajes no jugadores (PNJs) y en donde se encuentran los objetos.

En las salas que hemos descrito anteriormente hemos mencionado la existencia de tres objetos muy diferentes. Unas ramitas de tomillo, un cuchillo de cocina y una pluma de ave para escribir.

Veamos cómo son por dentro exactamente estos ficheros (véase el apéndice titulado **Cómo fabricar objetos**)

Tomillo

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE objetodeMine SYSTEM "mine_objeto.dtd">

<objeto version="1.0" autor="Aranarth">

<id>tomillo01</id>

<nombre>unas ramitas de [tomillo]</nombre>

<descripcion>
  <item>Unas pocas ramitas de tomillo muy oloroso.</item>
</descripcion>

<propiedades tipo="organico" categoria="comestible" aura="51" volumen="0.1"
peso="0.1" estado="85" valor="05" cargas="1" />

<usos>
    <uso tipo="comer">
<efecto atributo="vida" valor="+1" gasto="1"
    msg="Te comes el tomillo sin tratarlo"/>
    </uso>
    <uso tipo="usar">
<efecto atributo="aura_sala" valor="+1" duracion="300"
             gasto="1" msg="Un agradable aroma inunda el lugar."/>
    </uso>
</usos>

</objeto>
```

Cuchillo

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE objetodeMine SYSTEM "mine_objeto.dtd">

<objeto version="1.0" autor="Aranarth">

<id>cuchillo01</id>

<nombre>un cuchillo</nombre>

<descripcion>
  <item>Un cuchillo de cocina muy normalito.</item>
</descripcion>

<propiedades tipo="metal" categoria="arma" aura="50" volumen="0.2"
   peso="0.4" estado="67" valor="56"/>

<usos>
  <uso tipo="ponerse" posicion="mano_der">
    <efecto atributo="ataque" valor="+2" msg="Empuñas el cuchillo"/>
  </uso>
  <uso tipo="ponerse" posicion="mano_izq">

    <requisito atributo="destreza" valor="1" msg="Eres demasiado torpe
    para empuñarlo con la izquierda" />

    <efecto atributo="ataque" valor="+1" msg="Ahora puedes atacar con
    el cuchillo, pero con esa mano no lo manejas muy bien"/>

  </uso>
  <uso tipo="lanzar">

    <requisito atributo="destreza" valor="5" msg="Eres demasiado torpe
    para lanzarlo con la mano" />

    <efecto atributo="ataque" valor="+2" msg="Has lanzado el cuchillo!"/>

  </uso>

</usos>
</objeto>
```

Pluma de ave

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE objetodeMine SYSTEM "mine_objeto.dtd">

<objeto version="1.0" autor="Aranarth">

<id>pluma_de_ave01</id>

<nombre>una pluma de ave</nombre>
```

```
<descripcion>
  <item>Una pluma de ave para escribir.<item>
</descripcion>

<propiedades tipo="organico" categoria="accesorio" aura="50" volumen="0.1"
    peso="0.1" estado="90" valor="80"/>

<usos></usos>

</objeto>
```

Estos ficheros *NO* se guardan en el directorio **seroth** dentro del directorio **desc_salas.** Se almacenan todos juntos en el directorio **desc_objetos**.

2.5. El paso final. Pasear por nuestras salas

Existen varias formas de poder visitar nuestras salas o áreas pero aquí explicaremos la que, a nuestro entender, es más sencilla.

Para empezar, las salas han de estar localizadas en un directorio de área (en nuestro caso, **seroth**) y éste dentro del directorio **desc_salas.** Es necesario, por tanto, conseguir una copia de la versión de Minë que aparece en la página web del proyecto (http://eutherpe.org) concretamente, en la sección **jugar ya!**. Habréis de montar un servidor de Minë en vuestro propio pc para poder ver las salas. Tras haber seguido las instrucciones de la página, tendréis un Minë propio corriendo en vuestro ordenador y podréis conectaros a él con cualquier cliente MUD indicando que la IP del servidor Minë es **127.0.0.1** (una dirección IP especial que siempre se refiere a la máquina en la que estáis). El puerto, naturalmente, seguirá siendo **2000**. Cuando hayáis comprobado que podéis jugar a Minë en vuestro ordenador (sin necesidad de estar conectados a Internet), id al directorio **poblado** dentro del directorio **desc_salas** y modificad un fichero de sala cualquiera añadiendo una salida más que apunte a uno de los ficheros de vuestra área.

Por ejemplo, podréis poner algo así en el fichero de **poblado** que elijáis:

```
<sureste id='seroth-recibidor01'>el recibidor de la Casa de Seroth</sureste>
```

Y al llegar a la sala en cuestión veréis aparecer una nueva salida que os conducirá (a modo de portal interdimensional) a una de las salas de vuestra área.

Normalmente *NO* es necesario reiniciar Minë cada vez que se efectúan cambios en los ficheros de **desc_salas**, ya que estos se reconocen automáticamente.

Los ficheros de objetos, simplemente colocadlos en el directorio **desc_objetos**.

Es natural que se nos escapen algunas comillas o que hayamos cerrado mal un paréntesis del tipo '<>'. Cada vez que se produzca un error (Minë los detecta), la conexión al servidor se verá interrumpida (aparecerá un mensaje en el cliente MUD) y en la ventana en donde salen los mensajes del servidor se os mostrará el mensaje del error que provocó la desconexión. A veces será sencillo, tras leer las últimas líneas del error, entender qué salió mal (y qué ficheros hay que revisar). En otras ocasiones no lo será tanto. En estos casos, no despesperéis, revisadlo todo bien y si seguís sin encontrar el error (o errores) enviados a los desarrolladores principales (Aranarth pabloruiz@gnu.org y Hirunatan hirunatan@hammo.org) el directorio comprimido del área (o las salas sueltas que creáis que tienen el error).

También podéis enviarnos salas o áreas que deseéis que pasen a formar parte del servidor de pruebas de Minë y así permitir que otras personas puedan explorar lo que hayáis construido (no os olvidéis de enviar los ficheros de objetos conjuntamente con el fichero comprimido del área o áreas).

3 Contribuir con código

3.1. El código fuente de Minë

Ésta es la otra forma de ayudar al proyecto Minë. Puedes incorporarte al equipo de programadores y escribir código fuente del programa. Tenemos una lista de correo especialmente dedicada a tratar este tipo de temas (que a los usuarios normales no les importan en absoluto) ambar-dev@mail.freesoftware.fsf.org.

El código fuente de Minë es 100 % **python**. Utilizamos **python** como lenguaje de programación por motivos históricos (el fundador del proyecto, Pablo Ruiz Múzquiz comenzó a programar 'ambar' -precursor de Minë- en **python**) y por motivos prácticos (**python** resuelve con elegancia muchos de los problemas que aparecen en el diseño de un juego de estas características, es multiplataforma y la curva de aprendizaje es muy suave, lo que facilita nuevos colaboradores).

También utilizamos **XML** para los ficheros fuente de las salas, PNJs y objetos ya que apostamos por estándares asentados y versátiles.

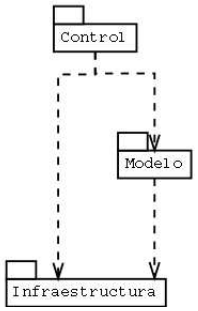
3.2. El mapa del código

Para no perdernos en el proceloso mar de líneas de código (actualmente los ficheros fuente de Minë ocupan cerca de 10.000 líneas), mantenemos a modo de mapa un diagrama de clases, hecho con UML. La última versión está disponible en la página electrónica de Minë, http://eutherpe.org/desarrollo.html.

Los diagramas muestran el diseño de alto nivel del programa, y son imprescindibles para conocer su estructura, antes de empezar a leer las clases de una en una. Hay tres grandes bloques:

- Modelo de datos $\longrightarrow$ una representación abstracta del mundo y sus habitantes, independiente del interfaz (debería poder usarse tal cual si cambiáramos el interfaz, por ejemplo a uno gráfico).
- Control $\longrightarrow$ el interfaz de usuario. Aquí se interactúa con los jugadores y se controla (o sea, se da órdenes) a los PJ y PNJ.
- Infraestructura $\longrightarrow$ algunas clases no relacionadas entre sí, con utilidades genéricas usadas por el resto de módulos.

Minë - Diagrama de bloques general



3.2.1. Modelo de datos

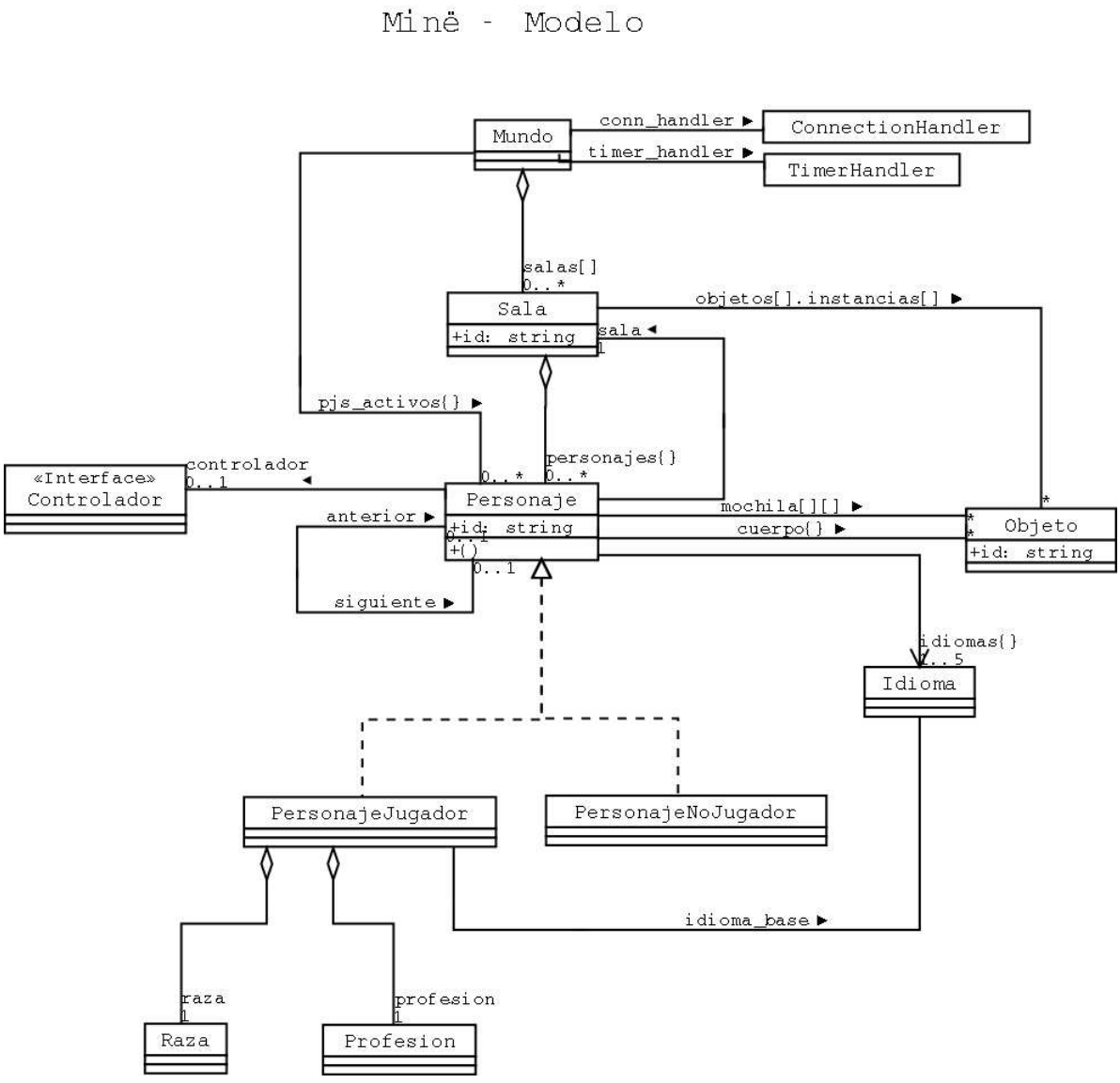
Éste es el bloque más significativo, que contiene el diseño *conceptual*. Cada clase de aquí representa una entidad del juego tal como lo ven los usuarios (personajes, salas, objetos...).

El diagrama se lee mejor de arriba a abajo. Al principio parece un poco confuso, pero todos los elementos tienen un sentido intuitivo.

El Mundo es único y representa eso, todo el mundo de Minë y lo que hay en él. Contiene una lista de Salas (el rombo en UML significa contener), y cada Sala, a su vez, contiene personajes: los personajes que se encuentran dentro de esa sala. Los Objetos ahora no están modelados del todo, pero en principio podrán estar dentro de una Sala o de un Personaje.

Un Personaje, a su vez, puede ser PersonajeJugador o PersonajeNoJugador (la línea de puntos con un triángulo, en UML, significa subclasificación). Todos los personajes conocen de 1 a 5 idiomas, y los jugadores tienen además una Raza y una Profesión.

Finalmente, todo Personaje es manejado por un objeto Controlador, que es quien le da las ordenes de lo que debe hacer, y a quien informa de las cosas que le ocurren. Controlador no es una clase, sino un interfaz, lo que significa que no contiene código, sino sólo la declaración de qué métodos debe

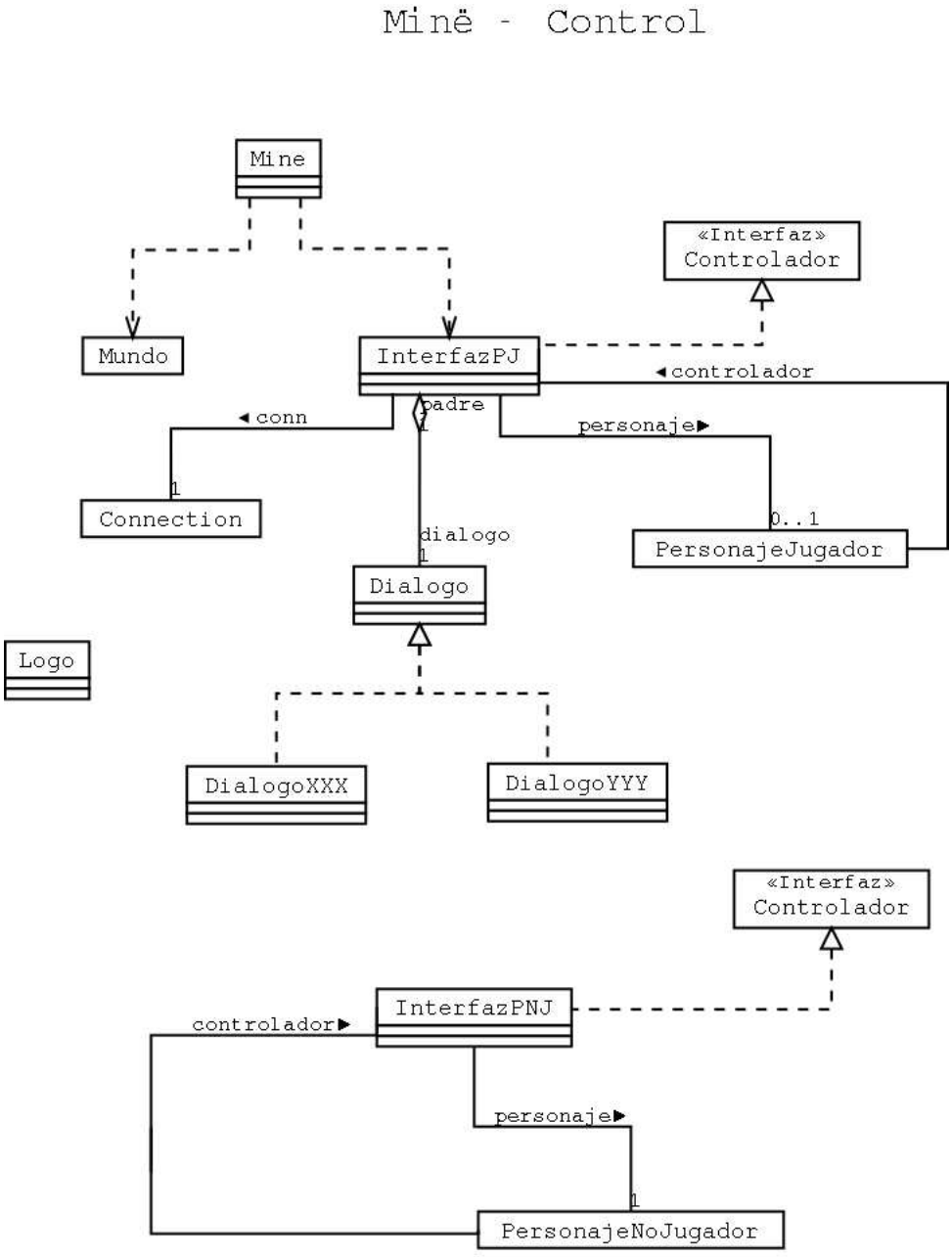

Minë - Modelo
Mundo
conn_handler
ConnectionHandler
timer_handler
TimerHandler
salas[]
0..*
Sala
+id: string
objetos[].instancias[]
sala
1
pjs_activos{}
personajes{}
0..*
«Interface»
Controlador
controlador
0..1
Personaje
+id: string
+()
anterior
siguiente
mochila[][]
cuerpo{}
Objeto
+id: string
idiomas{}
0..5
Idioma
PersonajeJugador
PersonajeNoJugador
idioma_base
raza
1
profesion
1
Raza
Profesion

tener una clase que pueda controlar Personajes. Las clases que implementan ese interfaz pertenecen al siguiente bloque del diagrama.

Hay algunas asociaciones más entre clases, que se pueden entender observando el código fuente. Las clases que tienen un único rectángulo (por ejemplo ConnectionHandler) no pertenecen a este diagrama, se muestran aquí sólo para poder ver las asociaciones.

3.2.2. Control.

Este bloque es un poco más abstracto. Contiene las clases que llevan el control del programa y la interacción con el jugador.



La clase Mine es el programa en sí. Es el fichero que se ejecuta al inicio y contiene el bucle principal. Inicializa el Mundo y los InterfacesPJ (las flechas punteadas en UML representan una asociación no permanente).

Un InterfazPJ es el objeto que controla a un PersonajeJugador. Acepta conexiones telnet para recibir comandos de un jugador y enviarle sus respuestas, y también da órdenes al personaje y recibe los "sucesos" o "eventos" que éste genera.

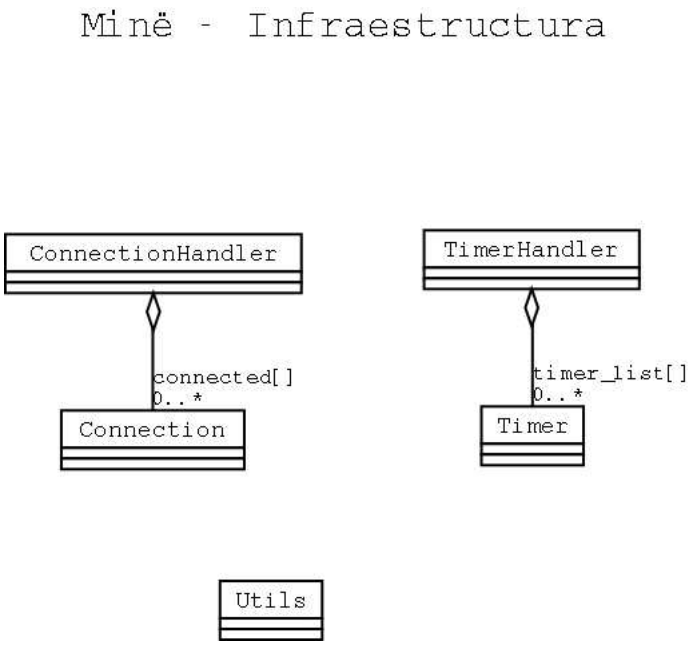
La funcionalidad de un InterfazPJ, al ser bastante complicada, está repartida en diferentes Diálogos. Cada Dialogo es una "máquina de estados" que representa una situación de diálogo con el jugador (por ejemplo, creación de un personaje nuevo, juego normal, modo combate, etc.).

Una máquina de estados es un objeto que tiene un "estado" en cada momento, y responde a "sucesos" que le ocurren bien cambiando a otro estado o realizando una acción. Por ejemplo, en el dialogo "crear un pj", al entrar en estado 1 pregunta el nombre del pj, y espera que ocurra algo. Si el jugador teclea un nombre correcto, lo registra y pasa al estado 2. Si teclea un nombre incorrecto (que ya existe, por ejemplo), devuelve un mensaje de error y continúa en estado 1.

Un InterfazPNJ tiene la misma misión, pero esta vez sobre un PersonajeNoJugador. La diferencia es que no tiene ningún socket telnet, sino que las órdenes a ejecutar las decide un programa de inteligencia artificial (más o menos "inteligente").

3.2.3. Infraestructura

En este bloque tenemos clases que representan conceptos genéricos que tienen más relación con el sistema operativo que con un juego MUD.



ConnectionHandler y Connection representan "sockets" TCP/IP, es decir, los canales de comunicación que implementan el protocolo telnet, usado para hablar con los jugadores.

TimerHandler y Timer son temporizadores, o cronómetros, útiles para medir tiempos. Utils es un módulo "cajón de sastre" para meter funciones sencillitas y variadas.

3.3. Entendiendo el código

Hemos seguido unas reglas básicas de codificación, para mantener un estilo coherente en distintos ficheros hechos por diferentes programadores. En primer lugar, tenemos un fichero fuente por cada clase, con el mismo nombre de la clase en minúsculas (p. ej. la clase `DialogoCrear` está en dialogo-crear.py). Todos los ficheros tienen al principio la nota de "CopyLeft", el comentario que indica su contenido, los import, y la clase. Todas las clases tienen un bloque de comentarios explicando la clase con detalle. Al final se incluye un trozo de código para pruebas unitarias del módulo.

Al escribir el código se siguen unas normas de diseño, que se pueden consultar en el apéndice titulado **Normas de diseño de código fuente de Minë**.

Al leer el código es fundamental leer las clases siguiendo el diagrama, de arriba a abajo, y leer y entender en primer lugar la cabecera donde se explica qué representa, y cuales son sus atributos.

El bucle principal del programa está en mine.py, pero casi no hace falta ni que lo mires. El programa se comporta como si todos los objetos del sistema estuvieran "vivos" y funcionando al mismo tiempo. Cuando ocurre cualquier cosa, el objeto correspondiente recibe un mensaje de "evento", es decir, "alguien" llama a un método de un objeto diciéndole que ha ocurrido algo (al principio no es necesario que sepas quién es ese alguien). Por ejemplo, si un jugador escribe algo desde su terminal, el objeto de tipo InterfazPJ correspondiente a ese jugador recibe una llamada al método `comando_recibido()`.

4 Apéndices

4.1. Descripción del sistema de conversaciones con los PNJ

Los Personajes No Jugadores tienen su importancia dentro de la ambientación del mundo de la Tierra Media. Son sus residentes naturales, los vecinos con los que puedes mantener una conversación formal pero agradable, y que te pueden revelar algún secreto de la región que estás visitando. A continuación se describe el mecanismo que se incluirá en Minë para hablar con ellos.

El sistema de conversación con los PNJ está ideado para facilitar a los jugadores las tareas de:

- Crear nuevas conversaciones PJ - PNJ por parte de los maestros de lugares.
- Interactuar con los PNJs durante el juego.

Otro objetivo importante es el permitir que los PNJs ayuden a crear un entorno de juego creíble y una historia interesante. La conversación que se puede sostener con un PNJ la creará un Maestro de Lugares, con un mecanismo similar al que se usa para crear habitaciones.

Cada conversación estará centrada en "temas" que tienen asociado un fragmento de diálogo. El jugador, en función de cómo transcurra el diálogo, escoge el siguiente tema por el que quiere preguntar. Normalmente escogerá alguno que se acabe de mencionar, aunque puede preguntar por alguno del que se haya hablado un rato antes, o que no haya salido en la conversación.

Los jugadores dispondrán de los comandos "+decir" y "+preguntar por". Con el comando +decir se supone que el PJ ofrece información que el PNJ no conoce, y con +preguntar (más frecuente) intenta sonsacarle. El estilo literario del texto puede hacer que se consiga un diálogo bastante bueno. Este tipo de conversación se podría adornar con verbos adicionales, sobre todo de comunicación no verbal (`+tocar, +abrazar, +pellizcar, +carraspear`, incluso `+besar`!) que pueden ayudar a cambiar el humor del PNJ.

Como muestra, éste es un fragmento del diálogo que se puede mantener con el "Tutor", un PNJ que aparecerá a la entrada del juego para enseñar a los jugadores nuevos los comandos de Minë:

```
    ? +decir hola
    - Saludos, buen mentor.
    - Bienvenido, mi joven aprendiz. ¿Estás preparado para comenzar tu instrucción sobre habilidades
de supervivencia en el mundo de la Tierra Media?
    - Asientes con admiración y respeto.
    - Para hablar conmigo, escribe "+preguntar por", seguido por el tema del que quieras hablar. Con
este comando puedes preguntarme sobre cualquier duda que tengas. También puedes escribir "+?" para
obtener una ayuda, con un resumen de todo lo que puedes hacer. ¿Has entendido bien?
    - Sí, maestro.
    - De acuerdo, vamos allá. Pregúntame sobre lo primero que quieras aprender.
    ? +preguntar por habilidades de supervivencia
    - Maestro, ¿qué es lo mas importante para sobrevivir en el mundo de la Tierra Media?
    - Lo primero que debes dominar es el funcionamiento de la línea de comandos. No irás muy lejos
si no sabes siquiera cómo desplazarte, ¿verdad? También es importante que sepas cómo combatir, y por
cierto que deberás aprender el arte de una buena conversación si quieres llegar a alguna parte.
    ? +preguntar por linea de comandos
    - Si escribes texto directamente, dirás en voz alta lo que has escrito, en idioma oestron. Puedes
escoger en qué idioma quieres hablar, de todos modos. En cambio, para realizar cualquier acción, primero
debes conocer el comando adecuado. Ya conoces "+preguntar por" y "+?". Los comandos empiezan por el
signo "+". Todo lo que escribas detrás de este signo es un comando que intentarás realizar.
    ? +preguntar por ejemplo
    - ¿Podrías darme algún ejemplo de manejo de la línea de comandos, mi buen mentor?
    El tutor se rasca la cabeza unos instantes, meditando la respuesta.
```

```
    - Si escribes: "Estoy de buen humor, ¿tú no?", dirás ese texto y te podrán oír todos los que estén
en la misma habitación. También puedes escribir el comando: "+preguntar por supervivencia" para comenzar
una conversación sobre ese tema.
    ? +preguntar por combate
    - Para atacar a un personaje, escribe "+atacar" y el nombre del personaje. Esto comenzará una
pelea. No te recomiendo que empieces un combate a muerte, sin embargo, hasta que llegues a ser un
gran guerrero.
    ? +preguntar por personaje
    - Maestro, ¿qué tipo de criaturas me puedo encontrar en mis viajes?
    - Es pronto para que te preocupes por eso. Más adelante podrás estudiar este asunto, si te interesa.
    ? +preguntar por mundo
    El tutor parece un poco desorientado por tu pregunta.
    - ¿Por qué este mundo se llama Tierra Media, por cierto?
    - ¡Ah! Se llenaría un tratado entero para responder con propiedad esa pregunta. ¿Por qué no me
pides que te lo explique... en otra ocasión?
```

4.1.1. Creación de conversaciones

Una conversación estará repartida en varios ficheros de "contexto"; en cada fichero estarán incluidos varios temas fuertemente relacionados. Puedes pensar que un contexto es como una habitación: mientras hables de temas parecidos permaneces en el mismo contexto, y si preguntas por algo no directamente relacionado, te mueves a otro contexto. El diálogo se debe diseñar como un edificio, lleno de habitaciones y recovecos por explorar. Un diálogo puede ser pequeño y confortable como la sala de estar de un hobbit, o profundo y extenso como el palacio de un Señor Enano.

Lo más importante a destacar es que un mismo tema puede aparecer en distintos contextos. Si el jugador pregunta por ese tema, obtendrá primero la respuesta contenida en el contexto actual, o la del contexto más cercano. También se puede partir el texto asociado a un tema en varios fragmentos sucesivos. De esta forma se consigue que se pueda preguntar varias veces por un mismo tema y obtener respuestas diferentes en cada ocasión, haciendo avanzar el diálogo.

Para que el escritor tenga control sobre qué texto se va a mostrar en un momento dado, dispondrá de al menos estos 2 mecanismos:

1. Aislamiento de los temas. Los temas estarán agrupados en contextos. Si el jugador cambia bruscamente de contexto, se activará la respuesta por defecto del tema solicitado; pero si se mantiene preguntando por tópicos del mismo contexto, se pueden ir activando respuestas preparadas más sutilmente, que formen una línea argumental de diálogo. El atributo de inteligencia del PJ debería permitir detectar, como si fuesen objetos ocultos, los temas de conversación más relacionados con el que se está diciendo ahora mismo. Esto representa que los personajes más inteligentes pueden encontrar más fácilmente los matices en la conversación con el PNJ, y aprenden qué es de lo que más le interesa hablar. Así, un jugador con un PJ inteligente podrá mantener mejores diálogos con los PNJs que uno tonto.

2. Estado previo del diálogo: memoria, humor del PNJ, y requisitos. No se debería repetir dos veces un mismo fragmento de texto. Se debe registrar si de este tema se hablado ya, y en ese caso mostrar un texto alternativo de entre varios. Al acceder a ciertos items de texto se pueden definir hitos en el diálogo, indicando que el jugador ha descubierto algo de información. Puede haber items ocultos que sólo se presenten si se ha alcanzado un hito concreto, o items que dependan de requisitos (características del jugador, presencia de algún objeto en la sala...). Algunos de estos hitos pueden representar distintos estados de ánimo del PNJ (alegre, miedoso, enfadado...) y estar almacenados en una variable que se cambie cuando el escritor quiera.

El siguiente es un ejemplo de cómo crear un contexto de conversación:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE contextodeconversacion SYSTEM "mine_conversacion.dtd">
<contexto version="1.0" autor="Hizgael" pnj="tutor" id="tutor-cortesia">
<tema nombre="saludos" accion="decir">
   <sinonimo>hola</sinonimo>
   <sinonimo>buenos días</sinonimo>
   <item>
     <pregunta>Saludos, buen mentor. </pregunta>
     <respuesta>Bienvenido, mi joven aprendiz. ¿Estás preparado para comenzar tu instrucción sobre
habilidades de supervivencia en el mundo de la Tierra Media? </respuesta>
     <entorno><tu>Asientes</tu><otros><pj/>asiente</otros>con admiración y respeto. </entorno>
     <respuesta>Para hablar conmigo, escribe "+preguntar por", seguido por el tema del que quieras
hablar. Con este comando puedes preguntarme sobre cualquier duda que tengas. También puedes escribir
"+?" para obtener una ayuda, con un resumen de todo lo que puedes hacer. ¿Has entendido bien? </respuesta>
     <pregunta>Sí, maestro. </pregunta>
     <respuesta>De acuerdo, vamos allá. Pregúntame sobre lo primero que quieras aprender. </respuesta>
     <relacionado contexto="tutor-topicos" nombre="supervivencia"/>
     <relacionado contexto="tutor-comandos" nombre="linea de comandos"/>
   </item>
   <item>
     <pregunta>¡Hola de nuevo!</pregunta>
     <respuesta>¿Cuántas veces me piensas saludar hoy, jovenzuelo?</respuesta></item>
</tema>
</contexto>
```

Este contexto contiene un sólo tema, con dos items de texto. El segundo ítem sólo aparecerá si vuelves a preguntar por el mismo tema, en este caso el tema "saludos". Se pueden definir sinónimos para detectar las distintas palabras que el jugador puede usar para referirse al tema.

Cada ítem puede tener asociado un conjunto de temas relacionados, que a pesar de estar en otro contexto tienen relación con lo que se dice en el texto. Si el personaje pregunta por un tema que no está en el mismo contexto ni está relacionado con los items utilizados, la pregunta está "fuera de contexto" y el personaje no jugador se extrañará por el cambio brusco del diálogo.

4.1.2. Confianza entre el PNJ y los PJs

Otra idea importante es la de que la conversación dependa del grado de confianza del PNJ con cada uno de los personajes de la sala. El PNJ tendrá un humor (o ’ánimo’) base, que al inicializarse depende de los personajes presentes en la sala. Por ejemplo, el posadero puede empezar la conversación de mal humor porque la taberna se le ha llenado de enanos (este posadero, ha salido un poco racista).

El humor cambiará a mejor o peor, como efectos por activar items de texto específicos. También bajará (¡mostrando mensajes de ambientación como efecto secundario!) si el PJ cambia constantemente entre temas no relacionados; esto favorece que los jugadores procuren mantenerse "dentro de contexto" y mejora mucho la apariencia de realidad del diálogo.

Además, si un PJ hace subir el humor, la confianza del PNJ en él aumenta, o si le estorba o dice algo raro, disminuye. La confianza inicial en cada personaje estará en proporción con su atributo de Carisma. Ciertos contextos o items de la conversación sólo serán accesibles si la confianza en el PJ está por encima de un umbral.

4.1.3. Salas llenas de gente

¿Cómo se comporta el sistema de conversaciones con varios jugadores acosando a preguntas al PNJ desde varios frentes? Los diálogos hacen mejor efecto cuando un solo jugador habla con el PNJ.

El efecto confianza puede ayudar a resolver este problema. El PNJ sólo tocará los textos más delicados (es decir, que requieren una secuenciación correcta), si los PJ hablan de uno en uno. Además, sólo se dirigirá en este caso al PJ con el que tenga más confianza. Esto aproxima la conversación a un diálogo entre dos personajes.

Otra idea para mejorar esto es que, si varios jugadores hablan a la vez, el PNJ sólo responda a aquél con el que ya estaba hablando, o bien con el que tenga más confianza. Esto último refuerza la ambientación de la característica Carisma: los PNJs tenderán a hablar con los PJ más carismáticos.

4.1.4. Iniciativa de los PNJ

El personaje no jugador puede tener su propia agenda de intereses. Al tocar un cierto tema, se puede despertar su curiosidad por algún otro relacionado. Si el jugador no pregunta por este otro tema posteriormente, el PNJ tomará la iniciativa y lo introducirá en la conversación (’¿Te he dicho ya que maté 80 orcos en la Batalla de los 5 Ejércitos?’), en cuanto el diálogo decaiga.

Además, el incluir temas con la iniciativa del PNJ puede valer para que reaccione a palabras clave de los diálogos entre los PJ, sin que resulte demasiado forzado. Por ejemplo, si el grupo de jugadores se pone a discutir a voz en grito sobre el Nigromante en medio de la posada, seguro que alguno de los aldeanos intervendrá en la conversación para recriminarles. Para hacer esto, no hace falta que el sistema sea capaz de entender el lenguaje natural de los mensajes entre jugadores; basta que reaccione a la palabra clave "Nigromante".

Este sistema puede forzar a que los jugadores hablen con el PNJ más interesante de la sala, si antes les había pasado desapercibido.

4.1.5. Efectos especiales durante la conversación

Los diálogos entre PJ y PNJ deberían estar plagados de ’frases de ambientación’ que reflejen la actitud que presentan los personajes frente a cómo se está desarrollando el diálogo; o que refuercen el efecto dramático de los pasajes más importantes. Por ejemplo, cuando Gandalf lee en el Concilio de Elrond el texto inscrito en el Único en Lengua Negra, todos los presentes en la sala se estremecen, y un velo de oscuridad parece cubrir el sol.

Estos ’trucos de ambientación’ son especialmente efectivos si se refieren a posturas y expresiones de los personajes. Si aparecen frecuentemente durante el diálogo, se evita el efecto de bustos parlantes, personajes entre los cuales sólo se intercambia texto. Al describir su actitud corporal, la ambientación se enriquece y la credibilidad aumenta.

Las frases de ambientación no necesariamente deben estar ligadas a un ítem de texto concreto; habrá un mecanismo para que la frase se pueda activar automáticamente en momentos claves del diálogo, por ejemplo cada vez que el humor del PNJ cambia, o bien cuando desciende por debajo de un cierto umbral.

En realidad, este mecanismo de efectos especiales debería estar disponible para ser utilizado fácilmente en todos los comandos del juego: uso de objetos, magia, movimiento, combate... La diferencia entre Petria y Minë puede estar en que, en el combate, en vez de ’El monstruo de la cueva te pega un zurriagazo. Estás malherido’ el mensaje que se muestre sea de tipo: ’El Trasgo Maloliente hace girar su bastón nudoso sobre su cabeza. Intentas esquivarlo echándote a un lado, pero dada la estrechez

de la sala no consigues apartarte lo bastante rápido y el garrotazo te alcanza en el cuerpo. ¡Aï! ¡Ese último golpe parece haberte dejado sin respiración!’

4.2. Cómo construir una sala

Abrimos un editor de textos cualquiera del que podamos estar seguros que será capaz de guardar en formato TXT simple. Nosotros recomendamos siempre el uso de software libre.

Para los que deseen saber qué es lo que escribirán exactamente, podemos decir que se trata de un fichero XML. Un fichero XML es un documento con un lenguaje interno que nos permite conocer luego los elementos que lo conforman. La sintaxis puede resultar algo extraña para un profano pero no es necesario entenderla para escribir correctamente un fichero de éstos correctamente.

4.2.1. Datos generales en toda sala

Nada más comenzar a escribir colocamos 'de regalo' estas dos líneas:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMine SYSTEM "mine_sala.dtd">
```

No importa qué sala vayas a hacer, estas dos líneas aseguran que nuestro programa pueda dar por válido o no tu fichero.

Una vez escritas esas dos líneas, ponemos:

```
<sala version="1.2" autor="Aranarth" area="moria" comentario="una sala de prueba">
```

en donde cada uno puede cambiar tanto el autor como el área pero no la version (¡ojo! NO poner tildes en `area="loquesea"`). Lo de `version="1.2"` dejadlo así, es una manera de que el programa no se haga líos según vayamos evolucionando el formato de este fichero. Cada vez que se haga un cambio aumentaremos el número de versión, y el programa se negará a cargar ficheros con una versión distinta a la que él entiende. Para no desaprovechar el trabajo realizado, normalmente proporcionaremos un *script* para convertir automáticamente los ficheros a las nuevas versiones (si es posible hacerlo). Esto lo decimos para que la gente no se lleve a engaño y crea que `version="1.2"` se refiere a la versión de su sala... Si deseáis realizar algún comentario, hacedlo en el campo -opcional- de `comentario=""` (ej: `comentario="Sala de pruebas, intento de tener una sala con personajes no jugadores"`). No confundir tampoco con la version del lenguaje XML de la primera línea del fichero.

Bien. Lo siguiente es bastante sencillo.

```
<id>moria-entrada01</id>
```

dentro de `<id></id>` habréis de poner un texto **que no contenga espacios** y que identifique de forma unívoca al fichero en donde se guarda la sala. Para ello, la regla es que escribáis

```
nombre_de_area-nombre_de_sala.
```

Es decir, dos partes separadas por un guión "-", la primera coincide con el nombre del área que hemos puesto arriba, y la segunda es un nombre que no se repite dentro de ese área. En cada nombre se pueden usar únicamente letras, números y el signo "_" para separar palabras.

Importante: el nombre del fichero que estáis editando será SIEMPRE el id de la sala + la coletilla '.xml'. Éste puede ser un buen momento para guardar el archivo con el nombre indicado.

Si existe un área llamada "poblado" y creéis que la posada del poblado se merece un área para ella sola, indicadlo escribiendo `poblado_posada` en el campo área de todas las salas que pertenezcan a

la posada del poblado. La parte del id de una sala que se refiere a la sala en sí también puede contener más de dos palabras pero siempre separadas por guiones bajos (a modo de espacios), como por ejemplo; `piso01_cuarto02`. Con lo que el id completo resultaría ser:

```
<id>poblado_posada-piso01_cuarto02</id>
```

Al principio puede costar un poco acostumbrarse pero con el tiempo veréis que es mucho más inteligente para llevar buena cuenta de las áreas y las salas.

- ejemplos válidos para id: `moria-sala07, moria-pasillo77, lorien_sur-estancia03, tharbad_oeste-posada03_piso02_cuarto01.`
- ejemplos no válidos: `lorienflet03` (no habéis puesto - entre área y nombre de sala), `moriamazarbul` (no habéis puesto - ni un número, aunque sea 01), `tharbad-cocina8` (el número debería tener al menos dos cifras).

Un área puede ser una construcción grande (+20 salas) o un pueblo, o una sección de un camino muy largo, etc. En general se apela al sentido común de la gente. No puede haber áreas de 5 salas porque sería fragmentar demasiado todo. Recomendamos que las áreas tengan un mínimo de 30 salas aunque hay casos particulares en donde podemos bajar a 20.

Ahora viene algo bastante fácil, el nombre 'normal' de la sala, lo que aparecerá como título del lugar a los visitantes.

```
<nombre>la entrada a Moria</nombre>
```

Intentad que se quede en menos de 6 palabras pero que sirva de referencia. El nombre debe contener un artículo y empezar por minúscula, ya que será mostrado de la forma "Estás en la entrada a Moria".

4.2.2. La descripción de la sala

Ahora nos encontramos algo más largo pero que es a la vez lo más importante, sin duda. La descripción de la estancia en sí.

```
<descripcion>
   <item>Una gran sala se aparece ante ti. Tanto el techo como las paredes son de color rojizo
oscuro y notas que la atmósfera es bastante húmeda debido, sin duda, a las filtraciones de agua del
cercano lago. Tres columnas impiden que la estancia se derrumbe.</item>
   <item dificultad='12'>Hace tiempo que nadie limpia por aquí.</item>
</descripcion>
```

Vayamos por partes. Lo primero que has de notar es que hemos escrito `<descripcion>`. Exacto, nos hemos metido en el 'modo descripción'. Es aquí donde se demuestra la habilidad descriptiva del Maestro de lugares porque ha de tener en cuenta los siguientes elementos.

1. La descripción ha de estar comprendida entre 3 y 10 líneas a ser posible.
2. Esta descripción ha de ser intemporal y no depender de ninguna futura circunstancia. Nada de 'un enorme orco de da la bienvenida' porque no sabemos cuánto tiempo va a durar ese pobre orco ahí (poco, lo más probable) y quedaría muy mal que la gente leyese algo que **no**

es verdad. **Tampoco** alusiones a **objetos** que se puedan coger son bienvenidas por el mismo motivo. Quizá en versiones posteriores de Minë esto pueda modificarse pero de momento no es posible. Entended que hemos de ir por partes.

3. Ha de provocar al viajante una sensación global del lugar.

Si leéis los dos 'items' dentro de `<descripcion></descripcion>` veréis que uno lleva un `dificultad='12'` y el otro no. ¿Por qué? Digamos que una sala **siempre** tiene un `<item>` sin dificultad que se le aparecerá a cualquier jugador que entre ahí mientras que aquellos `<item>` con dificultad aparecerán impresos sólo a aquellos personajes que superen una tirada interna de percepción del programa. Para que os hagáis una idea de a qué equivalen ciertos valores en la dificultad ahí va una tabla resumen orientativa.

Dificultad de los items descriptivos en una sala:

0	Para eso no pongas dificultad. Lo ve todo el mundo por muy cegato que esté.
10	Hombre, con un poco que se fije uno aparece.
20	Se podría decir que uno debe tener buen ojo para ver esto
30	La cosa se complica pero con mucha atención se acaba descubriendo.
40	Aquí aparece una barrera natural. Mucha gente no superará esta dificultad hasta que sume bastante experiencia.
50	Difícil aunque un elfo no tendría muchos problemas.
60	Aquí el humano corriente puede abandonar porque se dejaría los ojos antes que descubrir la descripción extra.
70	Aquí los elfos tienen ya problemas serios para ver algo.
80	Esta descripción se reserva para Istari en la práctica.
90	Gandalf tendría serios problemas.
100	Reservado a Maiar de alto linaje.

Para que os hagáis más idea aún, diremos que en igualdad de tiempo jugado, de un grupo de humano, elfo, orco, hobbit o enano en los primeros niveles tendríamos que un humano apenas pasaría del 30, el elfo lograría quizá llegar al 50 en contadas ocasiones, el orco como el humano o un poco más, el hobbit como el humano y el enano se queda en el 40.

Podéis poner tantos items descriptivos como queráis, con o sin dificultad pero intentad que el resultado final no sea excesivo o inundaréis de letras la pantalla del jugador (con la consiguiente desgana lectora de éste)

4.2.3. Las salidas en una sala

Importante: los siguientes datos se refieren a las posibles salidas. se recomienda que se relacionen items de descripción con alguna salida. De vez en cuando colocad el mismo valor de dificultad en aquellos `<item>` y salidas relacionados. Luego pondremos un ejemplo.

Esto es bastante simple pero hay que entenderlo bien porque es lo que relaciona esta sala con el resto de ellas.

```
<salidas>
   <oeste id="moria-cocina01">la cocina</oeste>
   <este dificultad='12' id="moria-salon01">el salon</este>
```

```
    <norte id="moria-cuartucho08">un cuartucho oscuro</norte>
    <sur id="moria-entrada01">la entrada</sur>
    <arriba id="moria-bano01">el baño</arriba>
    <abajo></abajo>
    <otro id="moria-tunel01" tipo_cierre="cerradura" datos_cierre="moria-llave_de_tunel01"
        mensaje_cerrado="una reja con un candado bloquea el paso">un estrecho [tunel]</otro>
</salidas>
```

Importante: También son válidas las salidas 'noreste, noroeste, sureste y suroeste'.

Analicemos esto detenidamente.

lo primero es escribir `<salidas>` (y lo último `</salidas>`). Entremedias, y en este caso, tenemos 7 -siete- posibles direcciones a las que ir. Escribe tan sólo aquellas que de verdad existan o déjalas vacías.

Por ej: `<abajo></abajo>` quiere decir que no hay nada hacia abajo. Para eso, simplemente no lo escribas.

Cojamos la primera salida de nuestro ejemplo:

```
<oeste id="moria-cocina01">la cocina</oeste>
```

`<oeste` <– esto indica que se refiere a la sala a la que entraremos si nos dirigimos al oeste (a la izquierda en un mapa) ¡Haced un dibujo antes!

`<oeste id="moria-cocina01"` <– el id es el identificador de la sala que hay en esa dirección. Coincide con el nombre del fichero (sin el .xml) que contiene la información, y con el párrafo <id>...</id>de dicho fichero.

`<oeste id="moria-cocina01">la cocina</oeste>` <– Escribid en medio algo descriptivo que pudiese dar una idea al personaje de lo que hay en esa dirección. Puede coincidir exactamente con el verdadero nombre de la sala al oeste o puede que no. Por ejemplo, si Kloin el enano está vagando por los túneles de Moria y llega a una bifurcación del camino oeste-este puede que el este aparezca como "Al este divisas un pasadizo oscuro"

```
<este id="moria-pasadizo87">un pasadizo oscuro</este>
```

y el oeste como "Al oeste divisas un pasadizo con [restos] humanos"

```
<oeste id="moria-pasadizo88">un pasadizo con [restos] humanos</oeste>
```

Si Kloin el enano decide ir hacia el oeste se encontrará con que ha entrado en "La Antesala de Mazarbul" que, desde lejos, parecía un simple 'pasadizo con restos humanos'.

Los jugadores también pueden usar la primera palabra de la descripción para moverse (exceptuando artículos: el, la, los, las, un, uno, una, unos, unas). En el ejemplo anterior es lo mismo decir "+sur" que "+entrada". Si no interesa la primera palabra, o hay varias salidas que repiten, se puede indicar otra rodeándola entre corchetes. En el caso que tenemos aquí, "+pasadizo" seria igual que "+este", y "+restos" seria "+oeste". Ojo, solo puede haber una palabra entre corchetes, sin espacios. Esto es imprescindible si se usan salidas con dirección <otro>.

Bien. también está la dificultad. Si creéis que una de las salidas no es visible a simple vista podéis poner un `dificultad="37"` que se aplicará de la misma forma que los items de la descripción de antes.

Un personaje que sea capaz de ver un ítem de descripción de `dificultad='35'` podrá ver cualquier salida con dificultad igual o menor a 35. Así que un *combo descripción-salida* puede ser el siguiente:

```
<item dificultad="26">Arriba alcanzas a ver una trampilla</item>
...
<arriba id="tharbad-herrero_cuartito03" dificultad="26">una trampilla</arriba>
```

cada uno en su respectivo sitio dentro del fichero, claro. No abuséis mucho de las `dificultad=' '`. Usadlo con mesura. Se trata de la Tierra Media, no de un laberinto oscuro. Una salida con dificultad por sala es una auténtica burrada. Una por cada diez salas, aceptable, una por cada 20, correcto. Por otro lado, un personaje incapaz de ver una salida, podrá cruzarla si sigue a otro que pueda.

Finalmente, tenemos los cierres. Las salidas normalmente se pueden transitar libremente, pero a veces nos podemos encontrar puertas con candado o algún otro tipo de obstáculo. De ello trata el `tipo_cierre`, que por el momento puede tener los siguientes valores:

- ninguno → éste significa que no hay cierre, es el que se sobreentiende si no ponemos nada.
- cerradura → cualquier tipo de cerradura o candado que se abra con una llave.
- horario → un cierre controlado por alguien que lo abre o cierra a determinadas horas. Cuando está cerrado, es posible que se tenga que abrir con una llave.
- magia → éste de momento no está hecho, pero en el futuro permitiremos bloquear mágicamente una puerta.

Si ponemos un tipo de cierre distinto de "ninguno", pondremos también datos_cierre, que sera:

- para `"cerradura"` → el id del objeto llave
- para `"horario"` → las horas en las que está abierto (sólo horas completas, de 0 a 24). Por ejemplo, "10-14,17-21" está abierto de las 10:00 a las 14:00 y de las 17:00 a las 21:00. Si hay una llave para abrirlo las horas en que está cerrado, ponemos su id al final, separado por un espacio. Por ejemplo, ”10-14,16-18 llave_gorda02”. OJO: es importante no poner ningún otro espacio en el texto.
- para `"magia"` → aquí pondremos el id del hechizo o algo así, cuando esté hecho.

Y no nos queda más que `mensaje_cerrado`, con el texto que le aparecerá al jugador si intenta pasar y no puede.

Ejemplos de cierre podrían ser:

```
   <norte id="xxxx" tipo_cierre="horario" datos_cierre="10-20" mensaje_cerrado="el guardia te dice
'el puente está cerrado, no se puede entrar en la ciudad hasta las 10'">

   <norte id="xxxx" tipo_cierre="magia" estado="cerrado" datos_cierre="bloquear_portal"
        mensaje_cerrado="hay una puerta fuertemente cerrada, no parece que haya medios de abrirla">
```

4.2.4. Propiedades globales de la sala

Ahora pasamos a una de las partes más sencillas, más rápidas y que más vida le dan al juego.

```
<propiedades tipo='caverna' subtipo ='entrada' volumen='25' luz='06' aura='56' combate="si"
       ocultabilidad='30'/>
```

A simple vista parece sencillo rellenar este campo. Y lo es, para qué negarlo.

Estas 'propiedades' se refieren a atributos muy específicos de la sala en la que estamos y la elección de los valores determinará mucho de lo que pueda acontecer en ella.

Vayamos uno por uno.

- tipo: Se refiere al campo semántico al que pertenece la sala. Puedes elegir entre: población, edificio, campo, agua, montaña y caverna.
- subtipo: No es obligatorio pero sí muy recomendable. Se trata de especificar más sobre el tipo de sala. A continuación mostramos una lista de los tipos y sus subtipos correspondientes:

Población *cielo abierto, suelo liso*	**Edificio** *techo, suelo liso*	**Campo** *cielo abierto, suelo irregular*	**Agua** *cielo abierto, suelo irregular, agua*	**Montaña** *cielo abierto, suelo irregular, mucha altura*	**Caverna** *techo, suelo irregular*
calle	salon	llanura	ribera	cima	entrada
callejon	habitacion	bosque	costa	ladera	cueva
plaza	pasillo	claro del bosque	acantilado	promontorio	pasadizo
portal	escaleras	colina	cala	grieta	sima
pasadizo	vestibulo	valle	rio	desfi ladero	rio subterraneo
puerto	cuarto sin ventanas	pantano	lago	quebrada	lago subterraneo
arco	sotano	paramo	mar	abismo	
embarcadero	calabozo	desierto	islote	cornisa	
azotea	alcantarilla				
	tejado				
	balcon				
	letrina				

- volumen: Se refiere, no al volumen físico sino a la capacidad de albergar a personas. Escribir un número entre 0 y lo que vuestra imaginación considere que es MUY grande. una persona normal ocupa 1 unidad de volumen (un hobbit 0.5 u.vol). Un dragón puede ocupar desde 30 hasta 500 u.vol. Si ponéis `volumen="0"` nadie podrá entrar. si ponéis `volumen="0.5"`, sólo los hobbits podrán entrar y sólo de uno en uno. Esto puede dar mucho juego.

  Una plaza del pueblo, si la consideráis una sala única, podría tener `volumen='200'` ó mucho más si se trata de Minas Tirith pero recomendamos que a partir de 200 subdividáis la estancia en varias o incluso convirtáis una zona con ambiente propio en un área diferente. Es decir, es mejor escribir '`minastirith_ciudad-plaza01_norte01`' `volumen='80'` '`minastirith_ciudad-plaza01_sur01`' `volumen='90'` que '`mintastirith_ciudad-plaza01`' `volumen='170'`. Siempre habrá excepciones, naturalmente, pero se apela al sentido común del 'maestro de lugares'. Un cuarto corriente estaría entre `volumen='6'` y `volumen='15'`.

  Si una sala está repleta no podrás acceder a ella. Te aparecerá un mensaje como 'la Cocina está llena de gente, no puedes pasar'.

- luz: esto que parece una tontería NO lo es. la luz tomará un valor entre 0 (oscuridad total) y 100 (¿el negro? ¿qué es el negro?). No cometáis el error de describir la sala como 'oscura y viscosa' y poner `luz='78'`.

  Al mediodía en la plaza mayor podríamos tener `luz='70'`. Por la noche, con luna llena, `luz='27'` En un cuarto con ventana, a media tarde, `luz='50'`. En los pasadizos de Moria, `luz='20'`

  Escribid el `luz=""` el máximo posible de luz atendiendo al mediodía. El programa, según qué hora sea, hará un pequeño ajuste automático.

La luz afecta a la percepción del personaje (vital para encontrar objetos, salidas o personajes escondidos), al combate (poca luz lo entorpece pero demasiada también), a los atributos de ciertas razas como los orcos por ejemplo, etc. Una `luz='90'` se considera mágica, al igual que una `luz='10'` se considera obra de algún tipo de hechicería oscura.

- aura: Llegamos a una de las propiedades más importantes de todas. El aura no es realmente algo espiritual pero sirve bien para hacernos una idea. El aura es la sensación de bondad o maldad que desprende una sala. Un 0 significaría encontrarse en las propias entrañas de Melkor y un 100 haber cantado con los Ainur el Ainulindalë. Los valores con los que se trabajará normalmente estarán en el rango [25-75] y sólo de forma MUY excepcional se debería poner un 85 (el mismo centro de Lórien) o un 15 (el trono de Sauron). El aura es algo que también tienen los objetos y los personajes, tanto jugadores como controlados por el ordenador. Esto quiere decir que un orco tendrá típicamente un valor de `aura='35'` mientras que un elfo, dependiendo de su casta podrá alcanzar `aura='70'`. El aura de la sala no sólo sirve para hacernos una idea del ambiente que se 'respira' en ella sino que puede afectar de forma directa a los personajes que se encuentren en ella y que posean auras muy diferentes a las de la sala. Un orco de `aura='30'` que entre en un santuario élfico de `aura='70'` verá disminuido su potencial a prácticamente la mitad o incluso puede morir con sólo tocar un objeto puesto que una diferencia de 40 puntos en aura es excesiva. Un camino perdido en mitad de la Tierra Media tendrá un `aura='50'`. Tened en cuenta en qué época estáis escribiendo vuestra área. Si lo hacéis a mediados de la tercera edad los valores son los que os hemos dicho pero según avanza el señor oscuro y llegamos al Señor de los Anillos no estaría de más restar 5 a todos los lugares normalitos.
- combate: puede tomar dos valores. 'si' o 'no'. en principio todo sitio permite el combate (así que si no pones, se sobreentiende que has puesto `combate='si'`) pero a veces conviene reservar ciertos lugares para la curación o para charlar sin ser molestados, o quizá es que no hay apenas sitio (`volumen='2'`) para hacerlo, y entonces podéis poner `combate="no"`.
- ocultabilidad: esto resume dos aspectos de la sala. por un lado lo abigarrada que está de trastos o lo llena de piedras y ramas, por otro la facilidad para ocultarse en ella. un valor de cero '0' indica que es imposible ocultarse por los motivos que sean mientras que un valor de 100 quiere decir que ni tú mismo serías capaz de encontrarte aunque te palparas. Un valor estándar para estancias en casas podría ser de `ocultabilidad='15'`. No convirtamos la Tierra Media en el juego del escondite, no te pases dando facilidades para ocultarse. Este valor también se tendrá en cuenta a la hora de intentar descubrir personajes escondidos en la sala. Naturalmente, el valor de la ocultabilidad se verá modificado dependiendo del número de personas que en esa sala ya se encuentran ocultas.

4.2.5. Objetos en la sala

Y llegamos a los objetos, otra de las partes más interesantes de una sala. En principio un objeto o lo tiene un personaje o lo tiene una sala. Cada `<objeto>` tiene un `id="identificativo"`, una `cantidad` mayor que 1 (si no se especifica se sobreentiende que es 1) y una `dificultad` de encontrarlo (se aplica el mismo criterio que arriba) aparte, naturalmente, de un breve texto o una palabra que haga referencia a él.

```
<objetos>
<objeto id="pocion01" cantidad="3">una poción recuperadora</objeto>
<objeto id="espada02" probabilidad="15" maximo="3">una espada brillante</objeto>
```

```
<objeto id="pocion02" cantidad="2" dificultad="14">una poción curativa</objeto>
</objetos>
```

El texto o palabra descriptivo (lo que se encuentra dentro de los signos > y <) no ha de ser necesariamente el texto 'verdadero' del objeto aunque sí tiene que ver algo con él.

Una espada herrumbrosa no debería hacer referencia al fichero `id="vino34"`, por ejemplo. Pero una poción élfica reconstituyente sí podría verse a simple vista como una 'poción vitalizante' que hace referencia oculta al fichero con `id="pocion_elfica02"`. Es decir, el personaje verá sólo aquellos objetos con una dificultad aceptable (si un objeto no dispone de dificultad se sobreentiende que se ve a simple vista) y de éstos, sólo la descripción y la cantidad. Si un personaje decide recoger un objeto, automáticamente descubrirá su nombre verdadero pero no sus efectos hasta que no lo examine (a no ser que ya lo sepa porque haya tenido previamente uno de ellos) , y aún así podría no conocerlo todo sobre el objeto en cuestión hasta que no se decida a usarlo. Además, la `probabilidad` marca la facilidad (de 1 a 100) de que aparezca *espontáneamente* ese objeto en la sala cuando entra un personaje (o de que desaparezca también espontáneamente). Esto sirve para lugares en donde de vez en cuando aparecen hierbas curativas o pequeños trozos de pan, etc. Hay que usar esto con mucho cuidado. Recordad que la cantidad marca el número inicial de regalo de esos objetos, una vez que se acaben se comenzará a aplicar la probabilidad que será 0 si no pusiste tal atributo. El `maximo` indica el tope de número de ese objeto que puede generarse en esa sala. Tened en cuenta que si hay una probabilidad alta de que aparezca un objeto al entrar un personaje, al cabo de unos días aquello podría ser imposible de transitar. Si colocamos un `maximo="3"` nos aseguramos que nunca se superará esa cantidad para ese objeto en particular y esa sala. Si no colocas un máximo, se sobreentiende el valor inicial que aparece en la sala. Si el máximo es 0, no hay límite y la cantidad puede crecer indefinidamente ¡cuidado!.

En cualquier caso, cada objeto tiene un fichero y tendrás que leerte el apéndice **Cómo fabricar un objeto** para entender del todo cómo funcionan.

La descripción del objeto sigue las mismas normas que el `<nombre>` de un objeto, tal como se explica en el apéndice mencionado.

4.2.6. Personajes no jugadores (en construcción)

Y por último nos encontramos con los encuentros de personajes no jugadores.

```
<encuentros>
<encuentro id="unicornio" probabilidad="20" dificultad="30">un unicornio</encuentro>
<encuentro id="orco" probabilidad="15" cantidad="2" maximo="4">un orco</encuentro>
</encuentros>
```

Un encuentro se compone de -`id`-, que es el fichero en donde residen sus datos, una -`probabilidad`- que indica la facilidad con la que puede aparecer 'mágicamente' en el lugar (de 0 a 100 siendo 0 imposible y 100 todo el rato que alguien entre en la sala), una -`dificultad`- que marca la fácil o difícil que es descubrir al encuentro en la sala y -`cantidad`- que indica el valor inicial del número de ese tipo de encuentros y el posterior número de unicornios u orcos que mágicamente pueden aparecer cuando entre un personaje en la sala. Si se omite la probabilidad, se sobreentiende 0 y si se omite la cantidad se sobreentiende 1. El máximo funciona igual que en los objetos. Luego está la descripción genérica para que sepáis cómo hacer referencia a ellos en vuestras acciones de compra-venta, charla o combate.

4.2.7. Finalmente...

Y por último.

```
</sala>
```

Guardad este fichero como uno de tipo xml.
ejemplo: moria-entrada01.xml
Bien, un texto completo quedaría tal que así:

————————————-

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMinë SYSTEM "mine_sala.dtd">
<sala version="1.1" autor="Hirunatan" area="moria" comentario="pruebas">
<id>moria-entrada01</id>
<nombre>la sala nueva</nombre>
<descripcion>
   <item>Una sala nueva se aparece ante ti. Tanto el techo como las paredes son de color rojizo
oscuro y notas que la atmósfera es bastante húmeda debido, sin duda, a las filtraciones de agua del
cercano lago. Tres columnas impiden que la estancia se derrumbe.</item>
   <item dificultad='12'>Hace tiempo que nadie limpia por aquí.</item>
</descripcion>
<salidas>
   <oeste id="moria-cocina01">la cocina</oeste>
   <este dificultad='12' id="moria-salon01">el salon</este>
   <norte id="moria-cuartucho01">un cuartucho oscuro</norte>
   <sur id="moria-entrada02">la entrada</sur>
   <arriba id="moria-bano01">el baño</arriba>
   <otro id="moria-tunel01" tipo_cierre="cerradura" datos_cierre="moria-llave_de_tunel01"
          mensaje_cerrado="una reja con un candado bloquea el paso">un estrecho [tunel]</otro>
</salidas>
<propiedades tipo='caverna' subtipo ='entrada' volumen='25' luz='56' aura='56' combate="si"
    ocultabilidad='30'/>
<objetos>
   <objeto id="pocion01" cantidad="3">una pocion recuperadora</objeto>
   <objeto id="espada02" probabilidad="20" maximo="3">una espada brillante</objeto>
   <objeto id="pocion02" cantidad="2" dificultad="14">una pocion curativa</objeto>
</objetos>
<encuentros>
   <encuentro id="unicornio" probabilidad="20">un unicornio</encuentro>
   <encuentro id="orco" probabilidad="15" dificultad="36" cantidad="3" maximo="4">un orco</encuentro>
</encuentros>
</sala>
```

————————————-

Lo que un personaje estándar vería, por ejemplo, sería algo como:

Estás en la sala nueva:

****

Una sala nueva se aparece ante ti. Tanto el techo como las paredes son de color rojizo oscuro y notas que la atmósfera es bastante húmeda debido, sin duda, a las filtraciones de agua del cercano lago. Tres columnas impiden que la estancia se derrumbe.

Hace tiempo que nadie limpia por aquí.
****

Al oeste ves la cocina
Al este ves el salón
Al norte ves un cuartucho oscuro
Al sur ves la entrada
Hacia arriba ves el baño
También ves un estrecho [tunel]

Aquí se encuentran:
un unicornio
un orco
Galran el elfo

Aquí hay:
una pocion recuperadora (3)
una espada brillante
una pocion curativa (2)
?
_____

4.2.8. Consejos

1. Sé coherente. primero escribe la descripción general, luego las especiales y luego las propiedades, asegúrate que el sentido común impera. Un baúl en un sitio con `luz="75"` y `ocultabilidad="2"` no puede tener `dificultad="70"` porque es que se ve a simple vista.

2. No hagas lugares generadores de monstruos. hacer algo como `<encuentro id="orco03" probabilidad="95" cantidad="10">orco enorme</encuentro>` sólo tiene sentido en mitad de Mordor.

3. Para los caminos o lugares cuya descripción es muy repetitiva no te comas la cabeza, copia varias veces el mismo fichero, retoca el nombre de ese trozo del camino y hacia adonde apuntan las salidas y poco más.

   ejemplo:

   Estás en un lugar del camino de Bree:

   ****

   Un trozo de camino igual que tantos otros. Algo de vegetación y surcos de ruedas en la blanda tierra.

   ****

   Al norte ves que el camino sigue.

   Al sur ves que el camino sigue.

   No encuentras nada.

   Aquí no hay nadie.

?

——

cuyo fichero sería algo como:

————————————-

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE saladeMinë SYSTEM "mine_sala.dtd">
<sala version="1.0" autor="Aranarth" area="camino_bree">
<id>camino_bree-camino34</id>
<nombre>un lugar del camino de Bree</nombre>
<descripcion>
   <item>Un trozo de camino igual que tantos otros. Algo de vegetación y surcos de
   ruedas en la blanda tierra.</item>
</descripcion>
<salidas>
   <norte id="bree-camino35">que el camino sigue</norte>
   <sur id="bree-camino33">que el camino sigue</sur>
</salidas>
<propiedades tipo=’campo’ subtipo =’llanura’ volumen=’30’ luz=’66’ aura=’58’ combate="si"
     ocultabilidad=’13’/>
<objetos></objetos>
<encuentros></encuentros>
</sala>
```

————————————

4. Intenta no forzar a que la gente escriba con tildes porque si bien los creadores de este juego defendemos la ortografía creemos que dificultaría el disfrute de gente con teclados americanos. Por lo tanto los objetos o cualquier encuentro o lo que sea que haya que escribir su nombre no debe tener ni tildes ni nuestra preciada eñe. En lugar de la eñe, escribid una simple ’n’.

5. Todos los atributos que sean `dificultad="3"` o `atributo="constitucion"` DEBEN tener comillas en los valores. Serán no válidos cosas como `aura=3` ó `volumen=34`, cuidad que las comillas son las mismas y no se os escapa una mezcla como `ocultabilidad=’3"`, por ejemplo.

6. Ningún `<id>` debe contener ’.xml’ al final. sólo el fichero en donde se guardan esos datos.

4.3. Cómo fabricar un objeto

Lo primero es abrir un editor de textos simple con un nombre del estilo de `abrigo03.xml` o `pocion_elfica11.xml`, etc.

4.3.1. Inicialmente

Escribimos estas dos líneas tal cual.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE objetodeMine SYSTEM "mine_objeto.dtd">
```

luego escribimos esta línea modificando en cada caso el autor. El comentario y el área son opcionales (el área raras veces se escribe ya que los objetos son móviles y no parece que tengan que pertenecer a un área concreta).

```
<objeto version="1.0" autor="Hirunatan" comentario="una prueba que se me ha ocurrido" area="poblado">
```

4.3.2. Datos generales

Ponemos `<id></id>` y entremedias el nombre del fichero que estamos editando y que corresponde al identificador del objeto SIN .xml.

```
<id>pocion01</id>
```

luego ponemos entre `<nombre>` y `</nombre>` una par o tres de palabras por el que queremos que se conozca nuestro objeto (es mucho mejor sin tildes)

```
<nombre>una pocion recuperadora</nombre>
```

El nombre del objeto normalmente empezará por un artículo, ya que en las listas de objetos aparecerá algo así: ”Aquí hay una pocion recuperadora”.

La primera palabra (que no sea artículo) del nombre servirá para que los jugadores se refieran a ese objeto cuando esté dentro de una lista. Por ejemplo, un jugador puede escribir `+beber pocion` para usar el objeto del ejemplo.

Si queremos usar otra palabra que no sea la primera, o varias palabras, las rodearemos por un par de corchetes [ ]. En el caso anterior, si queremos que el usuario tenga que escribir `+beber recuperadora`, escribiremos:

```
<nombre>una pocion [recuperadora]</nombre>
```

4.3.3. Descripción

Llega la descripción. ponemos `<descripcion>` y luego tantos `<item>`’s como queramos (mínimo 1). No hay que poner muchos, con uno basta, aunque si queremos poner algún ítem secreto podemos hacerlo especificando una `dificultad='nn'` como aparece en el ejemplo de abajo.

```
<descripcion>
<item>Se trata de un elixir fabricado a partir de hojas frescas de diversas plantas y que devuelve
un poco de vitalidad a la persona que la ingiere.</item>
```

```
<item dificultad="40">Devuelve hasta 15 pv. si se conoce Hierbas</item>
</descripcion>
```

4.3.4. Propiedades generales

Y llegamos a las propiedades generales. Para `aura` consúltese el Apéndice **Cómo construir una sala**. `volumen`: lo que ocupa. desde 0 (despreciable) hasta infinito. Una caja de zapatos ocupa `volumen='0.2'`. `volumen='1'` es como un humano adulto así que con esa referencia debería valer.

- tipo: madera, metal, cristal, piel, tejido, piedra, organico, cera, liquido o barro (de momento).
- categoria (no es obligatorio): mobiliario, arma, recipiente, joya, armadura, herramienta, vestimenta, escritura, recipiente, comestible o accesorio (de momento)
- peso: en kilos
- cargas: un valor numérico que indica cuántas veces se puede usar hasta que el objeto sea descartado. un valor de '0' quiere decir que siempre se puede usar y es típico de objetos permanentes.
- valor: en monedas de estaño. una barra de pan o una jarra de cerveza cuestan 1me. una espada corriente cuesta 8 monedas de plata. una hierba curativa muy rara cuesta, por ej, 4 monedas de oro.

  TABLA de precios (copiado de MERP) 10me = 1 moneda de cobre 10mc = 1 moneda de plata 10mp = 1 moneda de oro 100mo= 1 moneda de mithril

  Así, si algo cuesta 3 monedas de oro, en valor pon `valor="3000"`.
- estado: el estado en el que está el objeto. 100 quiere decir que está como nuevo (o lo que 'nuevo' pueda significar en cada caso) y si llega a '0' el objeto se pierde irremisiblemente. Para subir el valor de estado habrá que utilizar ciertas habilidades. Un objeto no tiene por qué comenzar con un estado="100", pero si empieza con estado="0" se romperá en cuanto alguien lo coja.

```
<propiedades tipo="metal" categoria="recipiente" aura="65" volumen="0" peso="0.05" cargas="1" valor="10"
estado="100"/>
```

4.3.5. Usos

Llamamos uso a una forma concreta de utilizar el objeto. Una espada puede empuñarse o arrojarse (dos usos), una botella puede beberse, arrojarse o empuñarse (tres usos) y así con muchos objetos.

La sintaxis de los usos va como sigue:

```
<usos>
   <uso tipo=″xxx″ posicion=″yyy″ gasto=″zzz″ msg=″descripción del uso″ duracion=″nnn″>
      <requisito atributo=″xxx″ valor=″nnn″ msg=″mensaje de error si el requisito no se satisface″/>
      ...
      <efecto atributo=″xxx″ valor=″+/- nnn″ duracion=″nnn″ msg=″Descripción del efecto″/>
      ...
   </uso>
   ...
```

```
</usos>
```

Como puede apreciarse, un objeto puede tener varios usos y cada uso puede tener ninguno o varios requisitos y ninguno o varios efectos. Los tipos posibles para el uso son: **comer**, **beber**, **poner** (ponerse una pieza de ropa, un collar, un anillo, etc), **lanzar** (algo que se arroja y golpea, causando daño) y **usar** (uso genérico, como en un artefacto mecánico). El uso "lanzar" puede ser obviado ya que todo objeto es *lanzable*. Minë calculará unos requisitos y efectos genéricos para el objeto cuando se desee lanzar. Sin embargo, Minë no realizará estos cálculos automáticos si encuentra un uso de tipo 'lanzar' explícito en el fichero descriptivo del objeto.

Las diferentes posiciones en las que un objeto se puede **poner** son: **cabeza, cuello, tronco, brazo-izq, mano-izq, brazo-der, mano-der, dos-manos, pierna-izq, pierna-der, pie-izq, pie-der** (para los otros usos no es necesario indicar posición). Naturalmente, un arma puede empuñarse tanto con la mano derecha como con la izquierda así que pondremos dos usos diferentes para las dos manos. Suponiendo que los PJs son todos diestros (aunque esto es fácilmente cambiable) tendremos que los requisitos de destreza con la mano izquierda serán mayores y el efecto en el combate será menor.

El **gasto** es el número de cargas que gasta este uso. 0 = no gasta. * = gasta todas las que resten. Si se cumplen los requisitos pero no hay suficientes cargas para un uso concreto Minë avisará al jugador.

La **duración** en el efecto se refiere al número de segundos que dura el efecto (pon 0 si es definitivo, o no pongas `duración=″`). El tiempo lo lleva el servidor de Minë así que será independiente de los ordenadores de los jugadores. En el momento en el que un uso activa una *cuenta atrás*, Minë se hace cargo del tiempo y avisará cuando el efecto haya pasado (si es menester). La duración en el uso se refiere al tiempo que existe ese uso con ese objeto. Esto permite añadir usos a objetos durante el juego mediante habilidades especiales (encender una antorcha, por ejemplo).

Los **requisitos** pueden pedir ciertos valores mínimos a los atributos: **Fuerza, Destreza, Constitución, Inteligencia, Sabiduría, Carisma, luz** y **Aura**. Los efectos incluyen estos atributos y añaden: **Vida, Ataque** y **Defensa**.

En nuestro ejemplo particular:

```
<usos>
   <uso tipo=″beber″ gasto=″1″ msg=″Bebes un elixir revitalizante″>
      <requisito atributo=″destreza″ valor=″3″ msg=″Eres demasiado torpe para beber la poción″/>
      <requisito atributo=″aura″ valor=″47″ msg=″Te quema la lengua. No puedes beberla″/>

      <efecto atributo=″vida″ valor=″+15″ msg=″Te sientes mucho mejor. Has recobrado parte de
tu vitalidad″/>

   </uso>
</usos>
```

Como vemos, este objeto requiere una mínima coordinación cabeza-mano (`destreza='3'`) y un aura aceptable (`aura='47'`). Si se cumplen estos requisitos, la vida subirá 15 puntos. Es posible que personajes con habilidades especiales puedan sacar más partido a esta poción.

Uniendo todos los pedazos de nuestro fichero descriptor del objeto pocion01, tenemos:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE objetodeMine SYSTEM "mine_objeto.dtd">
<objeto version="1.0" autor="Aranarth">
 <id>pocion01</id>
 <nombre>una pocion recuperadora</nombre>
 <descripcion>
```

```
      <item>Se trata de un elixir fabricado a partir de hojas frescas de diversas plantas y que devuelve
un poco de vitalidad a la persona que la ingiere.</item>
      <item dificultad="40">Devuelve hasta 15 pv.</item>
    </descripcion>
<propiedades aura="65" volumen="0" peso="0.05" cargas="1" valor="10" estado="100"/>
    <usos>
      <uso tipo=”beber” gasto=”1” msg=”Bebes un elixir revitalizante”>
         <requisito atributo=”destreza” valor=”3” msg=”Eres demasiado torpe para beber la poción”/>
         <requisito atributo=”aura” valor=”47” msg=”Te quema la lengua. No puedes beberla”/>

         <efecto atributo=”vida” valor=”+15” msg=”Te sientes mucho mejor. Has recobrado parte de
tu vitalidad”/>

      </uso>
    </usos>
   </objeto>
```

4.4. Normas de diseño de código fuente en Minë

- Es pérdida de tiempo escribir código para que haga cosas en seguida, y luego borrarlo y volverlo a escribir para que haga cosas bien. Esto sólo es válido para hacer prototipos de alguna técnica nueva que aún no se conoce.
- Para el resto del código, hay que diseñar primero el interfaz de las clases y luego el código. El interfaz consiste en las declaraciones de clase y de métodos, incluyendo los "doc comments".
- Dicho de otra manera, hay que escribir primero el "contrato" de la clase y luego la implementación del contrato (para saber qué significa esto, seguir leyendo). El contrato es la parte más "creativa" de escribir, si está bien hecho, la implementación es casi automática.

4.4.1. Diseño por contrato

Tipos Abstractos de Datos Según esta técnica, cada clase representa un Tipo Abstracto de Datos, en el sentido matemático: es decir, una pieza de información, con propiedades (o atributos), y operaciones que realizan cálculos sobre esas propiedades o cambian el valor de las mismas. El interfaz de la clase debe coincidir con el T.A.D., y el resto de la clase debe ser oculto (privado).

ejemplo una Sala es una habitación o lugar del juego. Sus propiedades son el id, autor, nombre, volumen, personajes que contiene, etc. Sus operaciones son crear una sala nueva, entrar y salir un personaje, etc.

En Python, el mapa entre T.A.D. y clase no está perfectamente definido, pero *grosso modo*, las propiedades corresponden a los atributos públicos, y las operaciones a las funciones miembro, públicas también. La parte privada de la clase no tiene reflejo en el T.A.D., y no debe ser conocida por ningún otro trozo de código.

ejemplo Sala.id, Sala.autor, etc. son atributos, y `Sala.__init__(), Sala.entrar_personaje()`, etc. son funciones miembro.

Requisitos de las propiedades: dominios e invariantes Las propiedades deben cumplir una serie de requisitos. Por un lado, el DOMINIO o conjunto de valores que puede tomar (es decir, el tipo: string, int, etc.).

ejemplo El dominio del id de una Sala son los números enteros (int), el del autor son las cadenas de caracteres (string), etc.

En Python, los tipos existen pero no son declarados (es un lenguaje dinámicamente tipado), e incluso una variable puede adoptar valores de distintos tipos. Esto puede ser cómodo en proyectos pequeños, pero en los medianos o grandes, hay que indicar los tipos en el interfaz, y para ello se pueden usar los strings de documentación. Estoy buscando una forma más o menos estándar de hacerlo, pero de momento lo haremos así (si un atributo puede tener distintos tipos, se comenta también, pero no es algo tan frecuente):

```
>class Sala:
>"""<descripción de la clase, una línea>
>
><descripción más larga>
```

```
>
>Contiene los siguientes atributos:
>- id (string): <descripción>
>- luz (int): <descripción>
>[...]
>"""
```

Por otro, las propiedades tienen restricciones o condiciones que deben cumplirse siempre. A estas condiciones se les denomina INVARIANTES. Una instancia cuyas propiedades cumplen todas las invariantes es una instancia válida. NO SE ADMITEN INSTANCIAS NO VÁLIDAS EN NINGÚN MOMENTO, excepto durante la creación de la instancia o la ejecución de una única de sus funciones miembro públicas. A la salida de cualquier función miembro pública, el código debe asegurar que ninguna invariante se ha roto, en ningún caso (esto no es necesario en las privadas).

ejemplo son invariantes `Sala.area != None, aura in range(0,100), n_jugadores >= 0, n_jugadores <= len(personajes)`, etc. El `__init__()` debe dar valores a todos los atributos que satisfagan todas las invariantes. No se admite, por ejemplo, crear una sala con area == None, y luego darle un valor más tarde. Sí se admite que el `__init__()` la inicialice a None y más tarde llame a una función que lee el área de un fichero y actualice el valor, siempre y cuando todo ello ocurra dentro de la ejecución de `__init__()`.

En Python no hay ningún mecanismo para expresar invariantes, así que lo haremos también en la documentación. También estoy buscando una forma estándar. De momento será como sigue. Durante la depuración, si se quiere verificar algunas invariantes, se puede usar la instrucción assert justo antes del return de todas las funciones (no hacerlo sistemáticamente con todas, porque puede cargar mucho).

```
>class Sala:
>"""[...]
>[...]
>Invariantes:
>id != None
>aura in range(0,100)
>n_jugadores <= len(personajes)
>[...]
>"""
>
>def entrar_personaje(self, personaje):
>[...]
>assert self.n_jugadores <= len(self.personajes)
>
>def salir_personaje(self, personaje):
>[...]
>assert self.n_jugadores <= len(self.personajes)
```

Requisitos de las operaciones: declaración, precondiciones y postcondiciones Las operaciones también tienen requisitos. Por un lado, está la DECLARACIÓN (el nombre y tipo de los parámetros, y el tipo de valor devuelto).

ejemplo, la operación "entrar personaje en una sala" tiene un parámetro que es la sala y otro que es el personaje que entra; sus tipos son las clases Sala y Personaje. No devuelve ningún valor.

En Python los parámetros vienen explícitos, y los tipos se pueden poner en el comentario, si no son obvios. Pueden ser obvios en los siguientes casos: si el nombre coincide con el de una clase, el tipo es esa clase; si coincide con el de un atributo de la clase actual, el tipo es el mismo que el del atributo; si el tipo es int o string, en muchos casos queda claro con sólo el nombre. Lo mismo con el valor de retorno.

```
>def entrar_personaje(self, personaje):
>"""[...]
>self(Sala) -- obvio, el self siempre está claro
>personaje(Personaje) -- obvio, mismo nombre que clase
>"""
>
>def mensaje_filtrado(self, mensaje, idioma, oyente):
>"""[...]
>mensaje(string) -- puede ser obvio, no hay clase mensaje así que
>será un string
>idioma(Idioma) -- obvio, nombre de clase
>oyente(Personaje) -- en principio no es obvio, pero se sabe
>por la descripción del método
>
>Devuelve string -- puede ser obvio también por la descripción
>"""
```

Las PRECONDICIONES son requisitos que hay que cumplir antes de llamar a una operación. Pueden referirse a los parámetros de la operación o a las propiedades de la clase. Su comprobación es responsabilidad del código llamante (lo puede hacer bien explícitamente, con algún if, o bien apoyándose en una invariante o postcondición). La función debe asumir que las precondiciones se cumplen, y no debe volverlas a chequear. Así se asegura que cada chequeo se realiza sólo una vez, y se evitan redundancias.

ejemplo, la sala tiene una invariante que dice que el aura es mayor que 0. Si el constructor lee los datos de la sala de un fichero, al leer el aura debe comprobar que cumple la invariante, y si no, dar un error y no crear la sala. A partir de ahí, todo el resto del código asume que el aura es mayor que 0, y se puede pasar como parámetro a una función que tenga esto como precondición.

En Python las precondiciones las colocaremos en el texto del comentario, de la siguiente manera. Durante la depuración, también se puede usar assert para comprobar algunas precondiciones, justo al principio de la función:

```
>def salir_personaje(self, personaje):
```

```
>"""[...]
>Requiere:
>personaje != None
>personaje in self.personajes.values()
>[...]
>"""
>assert personaje in self.personajes.values()
```

Las POSTCONDICIONES son requisitos que una operación se compromete a asegurar que se cumplen una vez que la misma haya terminado con éxito. Pueden referirse a las propiedades de la clase o al valor de retorno, si lo hay. El código llamante no debe volverlos a chequear. En cierta forma, las postcondiciones representan la información semántica de la operación (su utilidad prevista), frente a la información sintáctica de la declaración.

ejemplo, la operación salir personaje de una sala asegura que si se completa con éxito, el personaje ya no está dentro de la sala.

En Python, se colocan de forma similar a las precondiciones. Si se usa el assert, se puede poner justo antes de volver, o en el código llamante justo después de hacer la llamada:

```
>def salir_personaje(self, personaje):
>"""[...]
>Asegura:
>personaje not in self.personajes.values()
>"""
```

El contrato El conjunto de requisitos de propiedades y operaciones se conoce como el CONTRATO del T.A.D. Se llama así porque se asemeja a un contrato real: existe un proveedor (la clase), clientes (cualquiera que lea sus propiedades o use sus operaciones) y cláusulas (los requisitos).

Igual que en un contrato real, cada parte se compromete a cumplir su obligación, y hay un sistema para tratar los incumplimientos. El cliente se compromete a asegurar las precondiciones. Si no se puede cumplir alguna de ellas, no debe llamar a la operación en ningún caso. Una función no debe añadir código para protegerse contra precondiciones no cumplidas (puede permitirse incluso dejar el ordenador colgado o matar el programa si quiere), ya que nunca se va a dar el caso cuando el programa esté depurado (durante la depuración se puede usar assert).

El constructor de la clase debe asegurar todas las invariantes. Si no es posible cumplir alguna de ellas, debe lanzar una EXCEPCIÓN y abortar la creación de la instancia. En ese caso el cliente debe tirar la instancia a medias sin usarla.

Finalmente, las operaciones de la clase se comprometen a verificar tanto las invariantes como las postcondiciones. Si no pueden cumplir alguna, se debe lanzar una EXCEPCIÓN. En ese caso, puede que algunas postcondiciones estén cumplidas y otras no (no se puede asegurar), pero al menos la instancia tiene que seguir siendo válida y la operación tiene que restaurar todas las invariantes que puedan estar temporalmente deshabilitadas.

Las excepciones no son un mecanismo de control de flujo. Sólo deberían ocurrir en caso de incumplimiento de invariantes o postcondiciones, y en casos excepcionales (tipo disco duro lleno, datos corruptos, versiones incompatibles, etc.). Para situaciones que puedan ocurrir en una ejecución normal del programa (por ejemplo, el usuario introduce un nombre que no existe, se ha llegado al final

de una lista, etc.) es mejor usar condiciones explícitas (con if, while, etc.). Del mismo modo, las excepciones sólo se deben capturar en dos casos: cuando haya que restaurar alguna invariante, y luego seguir pasando la excepción hacia arriba, o cuando se llegue a un punto en el que se pueda volver a intentar la operación. En particular está prohibido capturar una excepción y luego hacer return normal, devolviendo un valor falso o que no cumple las postcondiciones.

4.5. Descripción formal de los ficheros de salas y objetos

Un DTD (Document Type Definition) es la plantilla que especifica de manera formal el contenido de un fichero XML, para que un programa lo pueda leer. Si no sabes XML, no es necesario que entiendas estas plantillas, son útiles para los colaboradores que quieran participar en la parte de código que trata los ficheros.

4.5.1. DTD de una sala

```
<!ELEMENT sala (id,nombre,descripcion,salidas,propiedades,objetos?,encuentros?)>
        <!ATTLIST sala
                version CDATA #REQUIRED
                autor  CDATA #REQUIRED
                area CDATA #REQUIRED
                comentario CDATA #IMPLIED>

        <!ELEMENT id (#PCDATA)>

        <!ELEMENT nombre (#PCDATA)>

        <!ELEMENT descripcion (item+)>
                <!ELEMENT item (#PCDATA)>
                        <!ATTLIST item
                               dificultad CDATA #IMPLIED>

        <!ELEMENT salidas (oeste*,este*,norte*,sur*,noroeste*,noreste*,suroeste*,sureste*,arr

                <!ELEMENT oeste (#PCDATA)>
                        <!ATTLIST oeste
                                id CDATA #REQUIRED
                                dificultad CDATA #IMPLIED
                                 tipo_cierre CDATA #IMPLIED
                                 estado CDATA #IMPLIED
                                 datos_cierre CDATA #IMPLIED
                                 mensaje_cerrado CDATA #IMPLIED>

                <!ELEMENT este (#PCDATA)>
                        <!ATTLIST este
                                  id CDATA #REQUIRED
                                  dificultad CDATA #IMPLIED
                                  tipo_cierre CDATA #IMPLIED
                                  estado CDATA #IMPLIED
                                  datos_cierre CDATA #IMPLIED
                                  mensaje_cerrado CDATA #IMPLIED>

                <!ELEMENT norte (#PCDATA)>
                        <!ATTLIST norte
```

```
                        id CDATA #REQUIRED
                        dificultad CDATA #IMPLIED
                        tipo_cierre CDATA #IMPLIED
                        estado CDATA #IMPLIED
                        datos_cierre CDATA #IMPLIED
                        mensaje_cerrado CDATA #IMPLIED>

<!ELEMENT sur (#PCDATA)>
        <!ATTLIST sur
                        id CDATA #REQUIRED
                        dificultad CDATA #IMPLIED
                        tipo_cierre CDATA #IMPLIED
                        estado CDATA #IMPLIED
                        datos_cierre CDATA #IMPLIED
                        mensaje_cerrado CDATA #IMPLIED>

<!ELEMENT noroeste (#PCDATA)>
        <!ATTLIST noroeste
                        id CDATA #REQUIRED
                        dificultad CDATA #IMPLIED
                        tipo_cierre CDATA #IMPLIED
                        estado CDATA #IMPLIED
                        datos_cierre CDATA #IMPLIED
                        mensaje_cerrado CDATA #IMPLIED>

<!ELEMENT noreste (#PCDATA)>
        <!ATTLIST noreste
                        id CDATA #REQUIRED
                        dificultad CDATA #IMPLIED
                        tipo_cierre CDATA #IMPLIED
                        estado CDATA #IMPLIED
                        datos_cierre CDATA #IMPLIED
                        mensaje_cerrado CDATA #IMPLIED>

<!ELEMENT suroeste (#PCDATA)>
        <!ATTLIST suroeste
                        id CDATA #REQUIRED
                        dificultad CDATA #IMPLIED
                        tipo_cierre CDATA #IMPLIED
                        estado CDATA #IMPLIED
                        datos_cierre CDATA #IMPLIED
                        mensaje_cerrado CDATA #IMPLIED>

<!ELEMENT sureste (#PCDATA)>
        <!ATTLIST sureste
                        id CDATA #REQUIRED
                        dificultad CDATA #IMPLIED
```

```
                        tipo_cierre CDATA #IMPLIED
                        estado CDATA #IMPLIED
                        datos_cierre CDATA #IMPLIED
                        mensaje_cerrado CDATA #IMPLIED>

        <!ELEMENT arriba (#PCDATA)>
                <!ATTLIST arriba
                        id CDATA #REQUIRED
                        dificultad CDATA #IMPLIED
                        tipo_cierre CDATA #IMPLIED
                        estado CDATA #IMPLIED
                        datos_cierre CDATA #IMPLIED
                        mensaje_cerrado CDATA #IMPLIED>

        <!ELEMENT abajo (#PCDATA)>
                <!ATTLIST abajo
                        id CDATA #REQUIRED
                        dificultad CDATA #IMPLIED
                        tipo_cierre CDATA #IMPLIED
                        estado CDATA #IMPLIED
                        datos_cierre CDATA #IMPLIED
                        mensaje_cerrado CDATA #IMPLIED>

        <!ELEMENT otro (#PCDATA)>
                <!ATTLIST otro
                        id CDATA #REQUIRED
                        dificultad CDATA #IMPLIED
                        tipo_cierre CDATA #IMPLIED
                        estado CDATA #IMPLIED
                        datos_cierre CDATA #IMPLIED
                        mensaje_cerrado CDATA #IMPLIED>

        <!ELEMENT propiedades (#PCDATA)>
                <!ATTLIST propiedades
                        volumen CDATA #REQUIRED
                        luz CDATA #REQUIRED
                        ocultabilidad CDATA #REQUIRED
                        tipo CDATA #IMPLIED
                        aura CDATA #IMPLIED
                        combate CDATA #IMPLIED
                        subtipo CDATA #IMPLIED>

<!ELEMENT objetos (objeto*)>
        <!ELEMENT objeto (#PCDATA)>
                <!ATTLIST objeto
                        id CDATA #REQUIRED
                        cantidad CDATA #REQUIRED
```

```
                              maximo CDATA #IMPLIED
                              probabilidad CDATA #IMPLIED
                              dificultad CDATA #IMPLIED>

        <!ELEMENT encuentros (encuentro*)>
                <!ELEMENT encuentro (#PCDATA)>
                        <!ATTLIST encuentro
                        id CDATA #REQUIRED
                        cantidad CDATA #REQUIRED
                        maximo CDATA #IMPLIED
                        probabilidad CDATA #IMPLIED
                        dificultad CDATA #IMPLIED>
```

4.5.2. DTD de un objeto

```
<!ELEMENT objeto (id,nombre,descripcion,propiedades,usos)>
      <!ATTLIST objeto
            version CDATA #REQUIRED
            autor CDATA #REQUIRED
            comentario  CDATA #IMPLIED
            area CDATA #IMPLIED>

      <!ELEMENT id (#PCDATA)>

      <!ELEMENT nombre (#PCDATA)>

      <!ELEMENT descripcion (item+)>
            <!ELEMENT item (#PCDATA)>
                  <!ATTLIST item
                        dificultad CDATA #IMPLIED>

      <!ELEMENT propiedades (#PCDATA)>
            <!ATTLIST propiedades
                  peso CDATA #REQUIRED
                  volumen CDATA #REQUIRED
                  aura CDATA #REQUIRED
                  valor CDATA #REQUIRED
                  tipo CDATA #REQUIRED
                  estado CDATA #IMPLIED
                  categoria CDATA #IMPLIED
                  cargas CDATA #IMPLIED>

      <!ELEMENT usos (uso*)>
            <!ELEMENT uso (requisito*,efecto*)>
                  <!ATTLIST uso
                        tipo CDATA #REQUIRED
```

```
      posicion CDATA #REQUIRED
      gasto CDATA #IMPLIED
      msg CDATA #IMPLIED
      aplicable CDATA #IMPLIED>
<!ELEMENT requisito (#PCDATA)>
      <!ATTLIST requisito
            atributo CDATA #REQUIRED
            valor CDATA #REQUIRED
            msg CDATA #REQUIRED>
<!ELEMENT efecto (#PCDATA)>
      <!ATTLIST efecto
            atributo CDATA #REQUIRED
            valor CDATA #REQUIRED
            msg CDATA #REQUIRED
            elemento CDATA #IMPLIED>
```