UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Creación de un mundo virtual e interfaz gráfica como plataforma de investigación de agentes/robots autónomos

PROYECTO FIN DE CARRERA

AUTOR: GONZALO HURTADO NUÑO

TUTORA: MARÍA MALFAZ

2009

AGRADECIMIENTOS

En primer lugar quiero agradecer a mi familia, padres y hermana, el apoyo y los ánimos constantes durante la elaboración del proyecto. En especial a mi padre por su labor en la corrección de errores.

A Irene y Avelina por su apoyo y amistad incondicional durante todo el periodo que han durado los estudios, así como el desarrollo del proyecto.

Un agradecimiento muy especial para Mª de los Ángeles Malfaz, tutora del proyecto, por sus consejos y sugerencias sin los cuales este proyecto no hubiera visto la luz.

En definitiva a todos aquellos que saben que se lo merecen.

ÍNDICE DEL DOCUMENTO

LISTA DE FIGURAS

LISTA DE TABLAS

Capítulo 1:
INTRODUCCIÓN

1 INTRODUCCIÓN

1.1 Motivación del proyecto

El presente proyecto se enmarca dentro de una línea de investigación llevada a cabo en el departamento de Ingeniería de Sistemas y Automática. El objetivo final de esta línea es el diseño de un sistema de toma de decisiones para un robot autónomo y social. Como paso previo a la implementación de ese sistema en un robot real, esta investigación va a ser desarrollada utilizando agentes virtuales, en lugar de robots. El agente vive en un mundo virtual donde existen objetos, los cuales necesita para sobrevivir, y otros agentes.

Esta línea de investigación se centra en el desarrollo de robots personales. Los robots personales son robots diseñados para estar en un entorno común con personas, para entretener o ayudar. Los robots personales podrían tener múltiples funciones como:

- Mascota.
- Vigilancia.
- Robots para trabajos domésticos.
- Entretenimiento.
- Ayudante personal.

Puede ocurrir también que un mismo robot personal sea completamente autónomo y que además tenga varias de las funciones previamente descritas, como la de entretenimiento y ayudante personal. Así se podría tener robots que fueran capaces de interactuar con humanos de una forma natural y que además participaran en la sociedad. De esta manera, los humanos interaccionarían con el robot como si ellos fueran compañeros, no supervisores del robot. Este tipo de robots personales se denominan robots sociales. Se puede definir un robot social como "un robot autónomo o semi-autónomo que interacciona y se comunica con humanos, siguiendo las normas de comportamiento esperadas por la gente con la que pretende interaccionar".

De acuerdo con algunos autores, para que un robot sea realmente autónomo, no sólo debe ser capaz de acciones inteligentes, sino que debe ser también auto-sostenible. Es decir, el robot debe ser capaz de reconocer sus propias necesidades como energía, fallo de algún componente y auto-preservación en general. Si surge un problema, el robot debe adaptar sus acciones y planes, para oponerse a continuar por un camino que le llevaría a la "muerte". El robot debe ser capaz de librarse por sí mismo de situaciones peligrosas, no sólo evitándolas a través de planes de alto nivel, sino que cuando las cosas salen mal debe responder de una manera rápida.

Por otro lado, en general, se puede definir un agente autónomo como un agente con objetivos y motivaciones, que tiene además alguna manera de evaluar los comportamientos en términos del entorno y de las propias motivaciones. Sus motivaciones son deseos o preferencias que pueden llevar a la generación y adopción de objetivos, siendo los objetivos situaciones que deben ser alcanzadas. Estos objetivos finales de un agente autónomo, o sus motivaciones, deben estar orientados a mantener el equilibrio interno del agente.

El aprendizaje es una habilidad importante para el agente autónomo ya que le dota de la plasticidad necesaria para ser independiente. Es necesario que el robot/agente aprenda mediante la interacción con el entorno cómo actuar para lograr mantener su equilibrio interno.

En este contexto, resulta muy útil y necesario el desarrollo de un entorno en el que se pueda probar las relaciones, las capacidades del robot o agente, los comportamientos, las interacciones con dicho entorno y las consecuencias o resultados. Una forma de conseguirlo es mediante la utilización de mundos virtuales como plataformas de experimentación. Es evidente que dichos mundos virtuales no pueden ser tan complejos como el mundo real, pero constituyen una herramienta de investigación extraordinariamente interesante.

1.2 Objetivos del Proyecto

El objetivo principal de este proyecto es facilitar las tareas de aprendizaje y experimentación de un robot o agente mediante la creación de un mundo virtual. En dicho entorno, el agente, de forma autónoma, se mueve con objeto de cubrir sus necesidades y relacionarse con otros agentes. Además es necesario registrar y visualizar los datos obtenidos de dichas vivencias.

Para ello, en este proyecto se presentan los siguientes objetivos:

- **Creación de un Mundo virtual** por donde se mueve el robot virtual o agente, formado por una serie de dependencias donde el agente realiza una serie de acciones encaminadas a satisfacer unas necesidades básicas. Ver capítulo 3.
- **Creación de una Base de Datos** donde se almacenen los datos sobre el estado del agente y las variables del entorno. Ver capítulo 4.
- **Creación de una Interfaz gráfica** que permita la visualización de los datos así como de su evolución. Ver capítulo 5.

1.3 Estructura del documento

El presente documento descriptivo del proyecto se ha estructurado en las siguientes partes principales:

- Primera parte donde se describe el Entorno (Capítulo 2).

  En este apartado se detalla, a modo de presentación general y con la única finalidad de que pueden entenderse las razones que motivan los desarrollos específicos de este proyecto, el marco general en el que se encuadra el proyecto, descripción del agente, su forma de interactuar con el mundo exterior y la forma en que aprende de sus experiencias.

- Segunda parte donde se describe la Plataforma Virtual (Capítulo 3).

  Bajo este título se desarrollan los apartados que constituyen el proyecto en sí:

  - *Mundo Virtual* (Capítulo 3). Donde se describe el mundo virtual donde el robot o agente realiza las vivencias que determinan su aprendizaje.

- *Bases de Datos* (Capítulo 4). En este punto se especifican las bases de datos donde se almacenan los parámetros asociados a las actividades, estados y refuerzos que contribuyen al aprendizaje del agente.
- *Interfaz Gráfica* (Capítulo 5). Es una interfaz que permite obtener gráficos sobre el estado y la evolución del agente para su posterior análisis y evaluación.

Finalmente se incluye un apartado de conclusiones y trabajos futuros (Capítulo 6).

Capítulo 2: ENTORNO

2 ENTORNO

2.1 General

En este apartado se describe el ámbito de trabajo en el que se sitúa el proyecto. Asimismo se describen las características que definen al propio agente o robot virtual, los estímulos, las motivaciones, estados y sistema de aprendizaje, que influyen en su comportamiento, así como el entorno en el que se desarrolla su "vida".

Aunque el contenido de este apartado no forma parte del proyecto en sí, se ha considerado imprescindible su inclusión con el fin de comprender los objetivos del proyecto, así como los motivos que han llevado a los desarrollos realizados en el mismo.
Dicho entorno puede verse descrito con más detalle en bibliografía relacionada con el tema, así como en otros proyectos y tesis doctorales (ver referencias).

La arquitectura del entorno donde se enmarca este proyecto es la representada en la Figura 2.1.

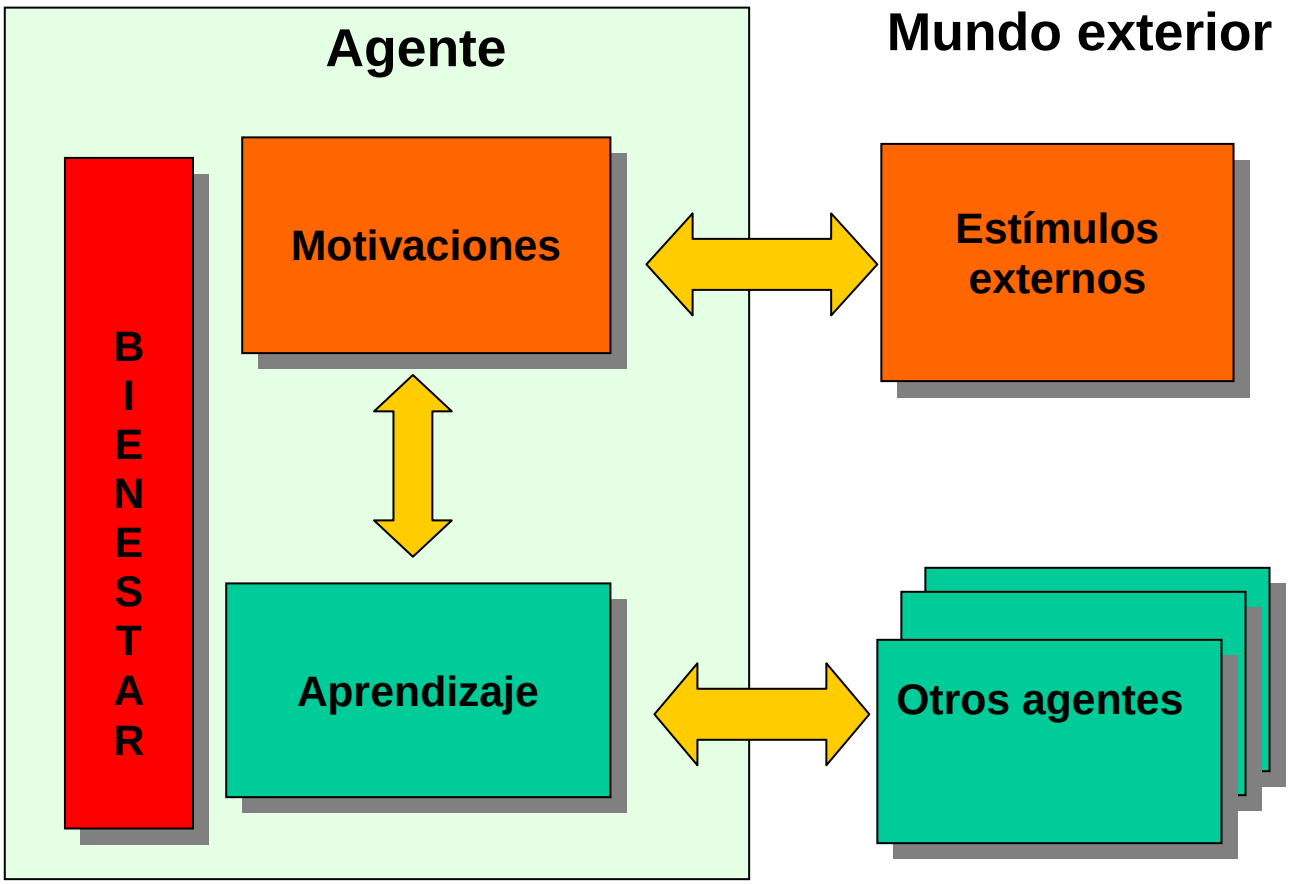

Figura 2.1: Entorno

Tal y como aparece reflejado en la figura 2.1, el agente dispone de dos módulos principales: un módulo motivacional y un módulo de aprendizaje. El primero hace que el agente realice determinadas acciones en el mundo exterior en función de las motivaciones dominantes, (comer, beber curarse, etc.). El módulo de aprendizaje le permite elegir su comportamiento en cada situación dependiendo de su estado interno de bienestar y del entorno en el que se mueve.

El robot o agente interactúa con el mundo exterior con dos tipos de objetos: estáticos, (comida agua, medicina), o con otro agente, lo que le permite aprender de su experiencia.

A lo largo de su vida, el agente irá desplazándose en busca de aquellos objetos que pueda necesitar para cubrir sus necesidades básicas y a la vez en busca de otros agentes con los que relacionarse socialmente.

2.2 Drives y motivaciones del agente

La palabra motivación deriva del latín *motus* e indica la raíz dinámica del comportamiento, es decir, aquellos factores internos más que externos que incitan a la acción. Las motivaciones del agente son los comportamientos dirigidos a satisfacer las necesidades, o *drives* del agente.

Los *drives* o necesidades implementados en este agente autónomo son:

- Energía
- Salud
- Sed
- Sociabilidad

Estos *drives* son los impulsores de un agente artificial, el cual vive en un mundo virtual. El agente tiene que comer (el drive Energía está relacionado con el hambre del agente) y beber para sobrevivir, así como curarse y relacionarse con otros jugadores. La necesidad de socializarse también está incluida como una de las necesidades internas de nuestro agente.

El valor inicial, de todos los *drives* está seleccionado como cero, que se considera como el valor óptimo. A cada paso del juego los valores de los drives son incrementados en ciertas cantidades, que dependen de la necesidad e importancia de cada uno.

Los *drives* no se incrementan de igual manera, por ejemplo, tal y como ocurre con los seres humanos, la necesidad de beber se produce antes que la necesidad de comer, que a su vez es más acuciante que la necesidad de tomar medicinas. En cuanto a la necesidad de relacionarse con otros agentes es la menor de todas y por tanto la que se incrementa más lentamente desde el valor inicial de cero.

Una variable significativa es la de "bienestar" que indica el estado general del agente. El bienestar del agente muestra el grado de satisfacción de sus necesidades y se calcula como:

$$Wb = W_{ideal} - (\alpha_1 D_{energía} + \alpha_2 D_{sed} + \alpha_3 D_{salud} + \alpha_4 D_{social}) \qquad (2.1)$$

Donde $\alpha_i > 0$ son los factores de personalidad que ponderan la importancia de cada drive en el bienestar del agente y D_i son los drives. $Wb_{ideal} > 0$ es el valor ideal del bienestar.

La intensidad de las motivaciones que mueven al agente a realizar determinadas acciones, (como comer, beber, curarse o relacionarse), depende del *drive*, (Energía, sed, salud, social), con el que están relacionadas así como de la existencia del estímulo externo correspondiente o incentivo.

La tabla 2.1 muestra dicha la relación entre motivaciones, *drives* y estímulos externos.

Motivación	Drive	Estímulo externo
Comer	Energía	Comida
Beber	Sed	Agua
Curarse	Salud	Medicina
Relacionarse	Social	Otro agente

Tabla 2.1 Tabla de Motivaciones

2.3 Estado del agente

En este sistema, el estado del agente es la combinación de su estado interno y su estado externo. El estado interno depende de su motivación dominante y el externo de su relación con los objetos.

El estado interno está definido como:

$$S_{interno} = [Hambriento, Sediento, Enfermo, Aburrido, OK] \quad (2.2)$$

En relación con los objetos, el agente puede estar en los siguientes estados;

$$S_{obj} = [Estar\_en\_posesión\_de, Estar\_cerca\_de, Saber\_donde\_hay] \quad (2.3)$$

Y en relación a otro agente:

$$S_{obj} = [Estar\_cerca\_de] \quad (2.4)$$

Todas estas variables son evaluadas como = [si, no]

2.4 Acciones del agente

El conjunto de acciones que el agente puede realizar en relación con los objetos externos, en función de su estado, son las siguientes:

$$A_{comida} = [Comer, Coger, Ir\ a, Explorar] \quad (2.5)$$

$$A_{agua} = [Beber\ Coger, Ir\ a, Explorar] \quad (2.6)$$

$$A_{medicina} = [Tomar\ medicina, Coger, Ir\ a, Explorar] \quad (2.7)$$

En cuanto a las acciones que puede realizar en relación con otros agentes con los que pueda interactuar son las siguientes:

$$A_{otro\ agente} = \begin{cases} Explorar \\ Robar\ comida/agua/medicina \\ Dar\ comida/agua/medicina \\ Saludar \\ No\ hacer\ nada \\ Pegar \end{cases} \quad (2.8)$$

Estos comportamientos afectan a los *drives*, reduciendo o aumentando su valor, lo que afecta al bienestar del agente, además de su estado emocional. En la tabla 2.2 pueden observarse dichas variaciones en función de la acción realizada por el agente.

Acción	Drive	Efecto
Comer comida	Energía	Reducir a cero
Beber agua	Sed	Reducir a cero
Tomar medicina	Salud	Reducir a cero
Que te saluden	Social	Reducir a cero
Que te roben algo	Social	Incrementar cierta cantidad
Que te den algo	Social	Reducir cierta cantidad
Que te peguen	Social	Incrementar cierta cantidad

Tabla 2.2 Tabla de Acciones, drives, efectos

La acción de “no hacer nada” no tiene ningún efecto en los drives de ninguno de los agentes que interactúan.

2.5 Sistema de aprendizaje

En este proyecto el sistema de aprendizaje de agente se basa en la utilización de algoritmos de aprendizaje por refuerzo, como el algoritmo Q-learning (Watkins, 1989), en el caso de que el agente interactúe con objetos estáticos únicamente. En el caso de una interacción social con otro agente se utilizará el algoritmo de aprendizaje por refuerzo para multi–agente. El algoritmo Q-learning proporciona una manera simple de aprender cómo actuar de forma óptima en dominios Markovianos. El agente aprende mediante el método de prueba y error.

El objetivo del aprendizaje por refuerzo es aprender el valor *Q(s,a)* a largo plazo que tiene estar en un estado. Esto representa el valor esperado del refuerzo recibido por el agente, debido a haber elegido la acción *a* en el estado *s*, llevándole a un nuevo estado, y actuar de forma óptima en adelante.

Por lo tanto el agente debe aprender los valores *Q(s,a)* para todas las parejas estado-acción. De manera que finalmente, el agente elegirá aquellas acciones que hagan máximo el refuerzo futuro.

El refuerzo utilizado en este sistema será la variación del bienestar sufrida por el agente como consecuencia de las acciones ejecutadas. De esta manera, se tiene una información bastante fiable sobre el efecto de las acciones sobre el bienestar.

Capítulo 3: PLATAFORMA EXPERIMENTAL

3 PLATAFORMA EXPERIMENTAL

3.1 Introducción

La plataforma experimental engloba los desarrollos realizados en el proyecto. Su objetivo es permitir el estudio del sistema de toma de decisión del agente mediante la utilización de un mundo virtual en el que se mueve, así como el almacenamiento de los datos obtenidos durante la exploración de dicho mundo y su representación gráfica para su análisis posterior.

La plataforma experimental consta de los siguientes elementos, ver Figura 3.1:

- **Mundo Virtual**
- **Base de Datos**
- **Interfaz Grafica**

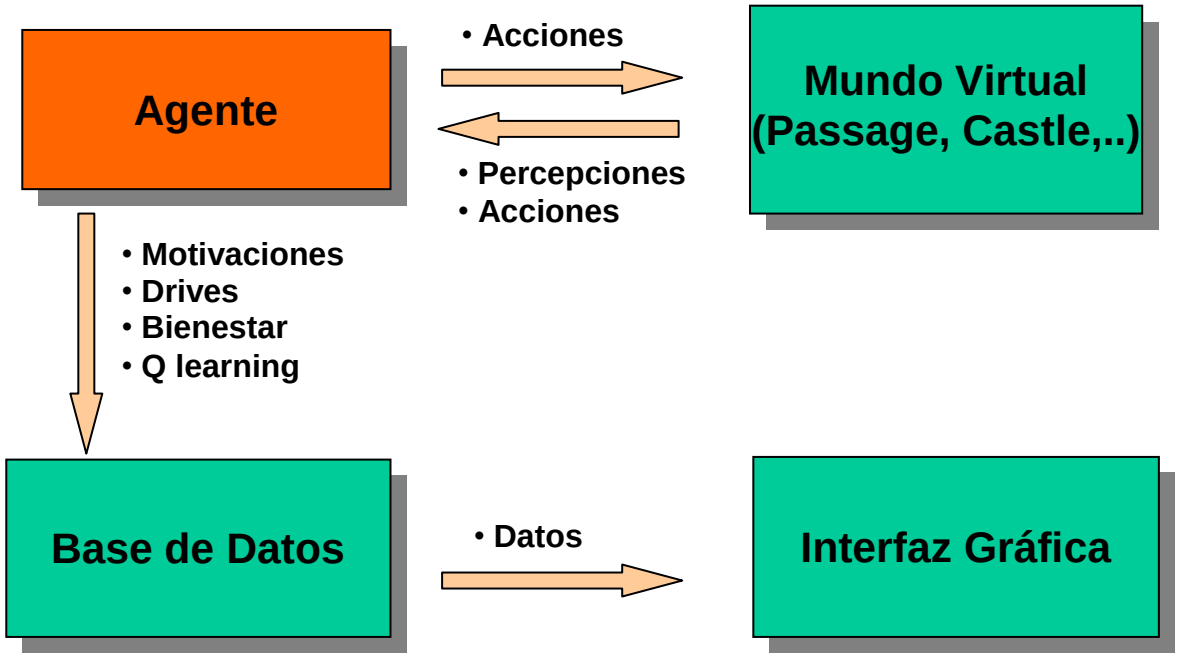

Figura 3.1 Plataforma experimental

La plataforma experimental está orientada a la experimentación del agente mediante la creación de mundos virtuales ("Passage" y "Castle"). Estas áreas están constituidas por habitaciones o dependencias donde el agente, motivado por una serie de necesidades primarias, (sed, energía, etc.) se ve impulsado a realizar una serie de acciones encaminadas a satisfacer dichas necesidades por medio de acciones (comer, beber, etc.). Todo ello se traduce en cambios en su bienestar.

Dichos cambios son almacenados en una base de datos relacional que consta de diversas tablas cuyo contenido se verá en el capítulo 4.

El desarrollo de una interfaz gráfica, capítulo 5, permitirá la representación de dichos datos para su estudio y obtención de conclusiones.

3.2 Mundo virtual

3.2.1 Estado del arte

Bellman en (Bellman, 2003) propone el uso de mundos virtuales como plataformas de experimentación para sus trabajos con agentes artificiales. Una de las cosas más importantes que se necesitan es un entorno en el que se puedan estudiar las relaciones entre objetivos, las capacidades del agente, los comportamientos, las interacciones con el entorno y las consecuencias o resultados en ese entorno. Con los mundos virtuales esto se puede conseguir, aunque por supuesto, estos mundos no son tan ricos como los mundos reales.

Los mundos virtuales surgen a partir de tres importantes líneas de desarrollo y experiencia:

- Juegos de rol en red y multi-usuario llamados MUVEs "Multi-user virtual environments" (Entornos virtuales multi-usuario).
- Entornos de realidad virtual y simulación distribuida avanzada, especialmente aquellas usadas en ejercicios de entrenamiento militar.
- Entornos de computación distribuidos, incluyendo internet.

El uso de estos mundos virtuales como plataformas de experimentación se está extendiendo mucho por la comunidad robótica y de inteligencia artificial. Por ejemplo en (Isbell et al., 2001) se presentan los estudios de aprendizaje por refuerzo de un agente artificial, el cual "vive" en un entorno multi-usuario llamado LambdaMOO.

Este entorno es uno de los más antiguos juegos de rol multi-usuario basados en texto, que está formado por habitaciones interconectadas, pobladas de usuarios y objetos que se pueden mover de habitación en habitación. Los mecanismos de interacción social son diseñados para reforzar la ilusión de que el usuario está presente en el espacio virtual. Otro ejemplo del uso de juegos de ordenador como plataforma de experimentación es el trabajo presentado en (Thomaz and Breazeal, 2006), en el que los jugadores interactivamente entrenan a un robot virtual para realizar una tarea. De la misma forma que en LabdaMOO, es un jugador externo el que da el refuerzo al agente para que aprenda a realizar una tarea en concreto o a sobrevivir.

El uso de estos mundos virtuales es cada vez más común tanto en el desarrollo de robots como en el campo de la inteligencia artificial.

3.3 Coffeemud. Creación del mundo. Passage y Castle

Como ya se ha dicho, los mundos virtuales son muy útiles como plataformas experimentales de aprendizaje. El objetivo final es el de dar una plataforma donde el agente puede aprender a comportarse para mantener sus necesidades satisfechas.

Los mencionados mundos virtuales suelen desarrollarse como juegos de rol multiusuario, basados en texto. En el caso de este proyecto se ha basado en la utilización de MUD

(Dominios Multi Usuarios), con el objetivo de crear un entorno complejo de experimentación y aprendizaje del agente.

Los MUDs "Multi-user Domains" (Dominios multi-usuarios) se han desarrollado a partir de 1979 y se refieren a juegos multi-usuarios basados en texto y aplicados fundamentalmente a juegos fantásticos del tipo "Dragones y Mazmorras".

La razón por la que se ha elegido la opción de un juego basado en texto en lugar de utilizar otras opciones más modernas que podrían disponer de mejores y más atractivas interfaces visuales, es la sencillez para enviar y recibir información del juego.

En un MUD típico, una persona se conecta a un servidor MUD utilizando un cliente Telnet y juega. En este caso se han desarrollado varios programas en C, quienes se conectan a un servidor simulando la existencia de varios jugadores, quienes se comportan según se ha especificado en el apartado 3, (Entorno). En este sentido un MUD ofrece la forma ideal de crear escenarios virtuales (áreas) con todos los objetos que se necesiten (comida, agua, medicinas) con los que el agente deberá interactuar.

Esto se realiza mediante la interacción con objetos, (comida, agua, medicinas,..), la comunicación con el juego se traduce en leer texto o enviar texto. Mediante texto se puede describir el contenido de las dependencias que configuran el mundo virtual (por ejemplo : hay comida en el centro de la habitación), y se pueden definir acciones que realiza el agente (moverse, coger comida,..).

Coffee MUD

Entre diferentes códigos de MUD's disponibles en la red, se ha elegido para la implementación de este proyecto el denominado Coffee Mud, basado en Java, por estar perfectamente documentado el software y disponer de manuales de uso claros.

Coffee MUD es un MUD cuyo código base fue escrito por Bo Zimmerman. Está basado en Java y es de uso libre bajo licencia Apache 2.0.

Coffee MUD tiene una lista muy activa de personas que colaboran en su continuo desarrollo.
Está escrito al 100 % en Java. Soporta cualquier base de datos JDBC/OBDC y ha sido probado utilizando MySQL, HSQLDB y Access/SQL server sobre ODBC.

Funciones Generales

- Soporta colores ANSI, Sonido MSP, compresión MCCP y etiquetas MXP.

- Código base estable y ampliable mediante opciones INI, Java y Java Script, incluso durante el tiempo de ejecución.

- Construido sobre un servidor web para ejecutar funciones de mantenimiento, administración trabajo en las áreas y acceso a información.

- Construido sobre servidor SMTP para e-mails de jugadores anónimos, listas de correo, anuncios y envío de contraseñas (passwords).

- Uso de bases de datos JDBC/ODBC.

- 7 claves básicas, 38 subclases, 6 familias de jugadores y más de 1200 nombres, sonidos, habilidades y aptitudes.

- Más de 700 aptitudes sociales.

- Varias posibilidades de escritura del constructor, incluyendo MOBPROG y Javascript estándar.

- Herramienta de edición visual on line basada en web.

- Sistema de seguridad personalizable. Asignación de privilegios por jugador.

- Capacidad para importar ficheros del área CoffeeMUD.

- Soporte MUD bajo demanda.

Áreas y habitáculos

- Área de creación on-line.

- Área de climas, estaciones y cambios de tiempo que afecten al juego.

- Diferentes tipos de terreno, incluyendo áreas sumergidas.

- Salidas según puntos cardinales y muchos tipos de puertas, portales, cortinas, etc.

Adicionalmente existe una gran diversidad de funciones relacionadas con el tipo de jugadores, armas, lenguaje propio de los juegos, etc., que no se describen por no ser significativas para este proyecto.

MUD Grinder

Para la creación y edición de los ambientes donde se desarrolla la acción, CoffeeMUD dispone de un editor de área y herramienta administrativa denominado MUD Grinder, el cual se apoya en una serie de pantallas de ayuda que permiten la creación y edición de los elementos que configuran el mundo virtual o "Passage", tanto los personajes como los habitáculos o dependencias que lo forman.

Asimismo dispone de una serie de pantallas que permiten visualizar la situación de cada jugador, resumen de estados, así como funciones estadísticas en función de diversos parámetros como: raza, clase de jugador, etc.

En resumen, el MUD Grinder es un editor de área GUI de Coffee MUD y una herramienta administrativa. Es accesible en el servidor de Coffee MUD a través de Internet. Esto significa que se puede usar un navegador web para administrar y construir un MUD desde cualquier lugar.

3.4 Descripción general de las funciones de creación y edición

A continuación se describen de forma resumida las funciones mencionadas de edición y administración, así como las pantallas en las que se apoyan dichas funciones.

Editor de Área

Las pantallas que ayudan a editar los componentes del Passage son las siguientes:

Mapa de edición de Área GUI

En la figura 3.2 se muestra una parte de un GUI Área Editor Map del Mud Grinder. Pinchando en una dirección gráfica se crea una nueva salida o una nueva dependencia (Habitación). Las direcciones más oscuras son salidas ya existentes. Las puertas son claramente indicadas por caminos en las líneas verticales. Pinchando en una caja de habitación se permite editar el habitáculo o habitación.

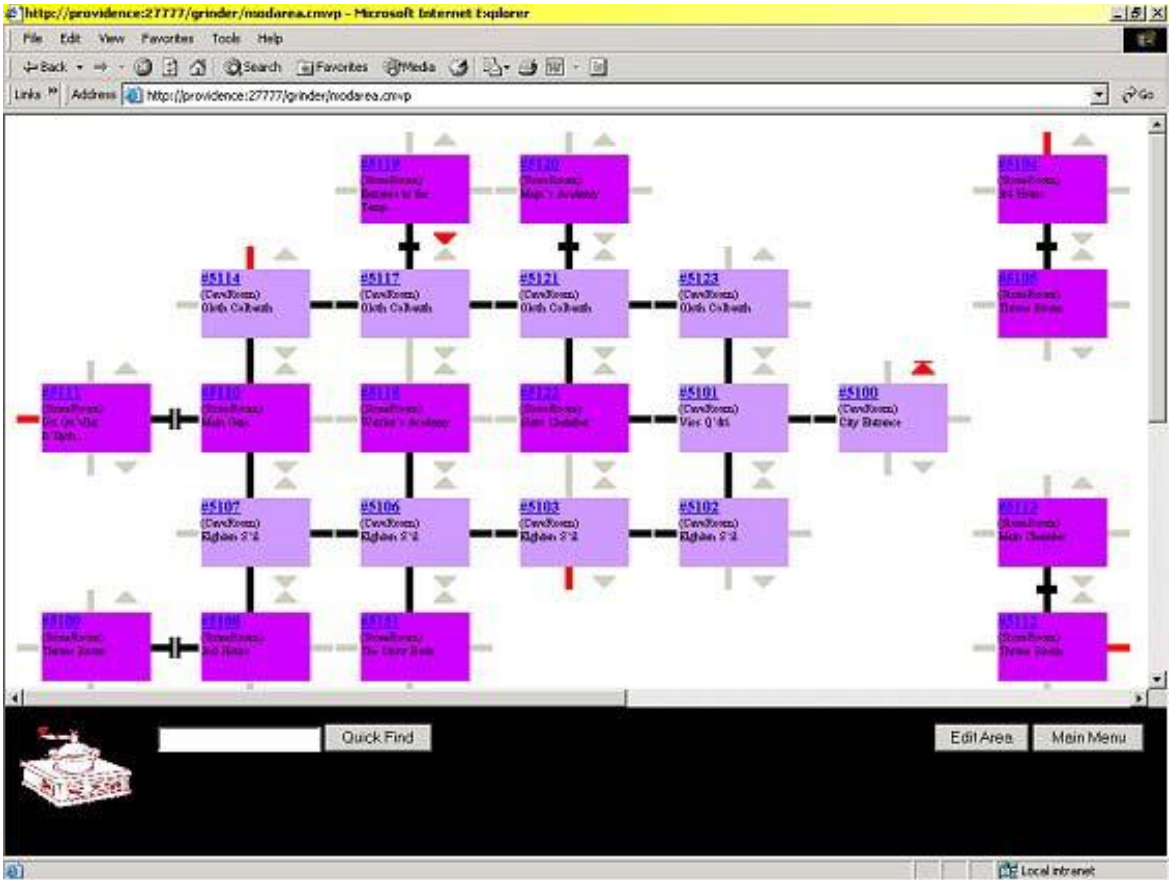

Figura 3.2 Editor de área

Pantalla editora de dependencias (habitaciones)

En la figura 3.3 se muestra el editor de habitáculos o dependencias. Este editor de habitáculos permite fácilmente escribir descripciones, añadir personajes y detalles, o cambiar las propiedades y comportamientos del habitáculo o habitación, simplemente pinchando con el ratón.

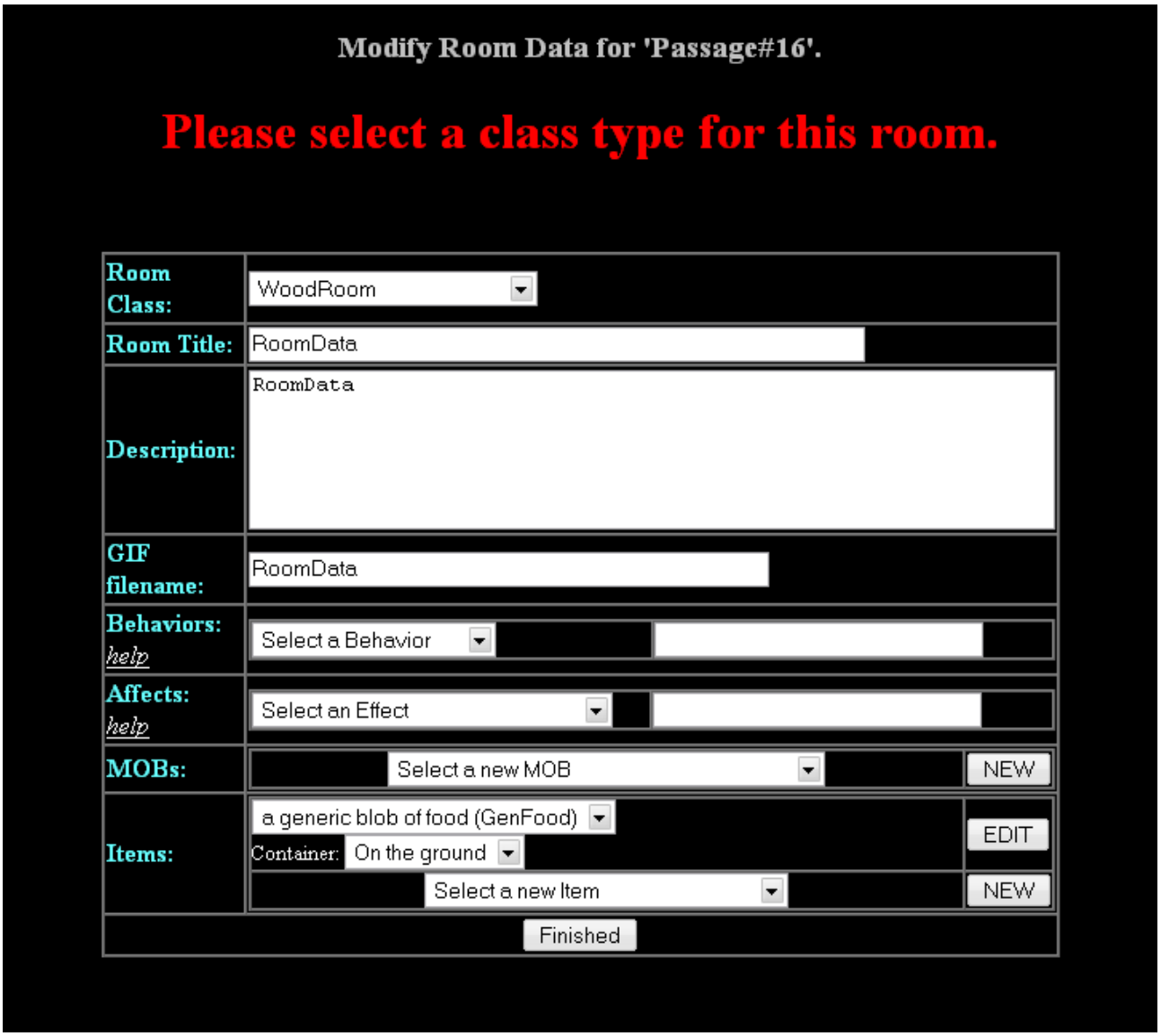

Figura 3.3 Editor de dependencias

Editor de Objetos (Item Editor)

En la figura 3.4 se muestra la pantalla del editor de objetos. Nuevos objetos (ítem en inglés) se crean y modifican de la misma forma desde las pantallas del Editor de habitáculos o del editor MOB. Hay marcadores de los diferentes tipos de detalles, cada uno de ellos con diferentes propiedades, que se pueden modificar desde la pantalla.

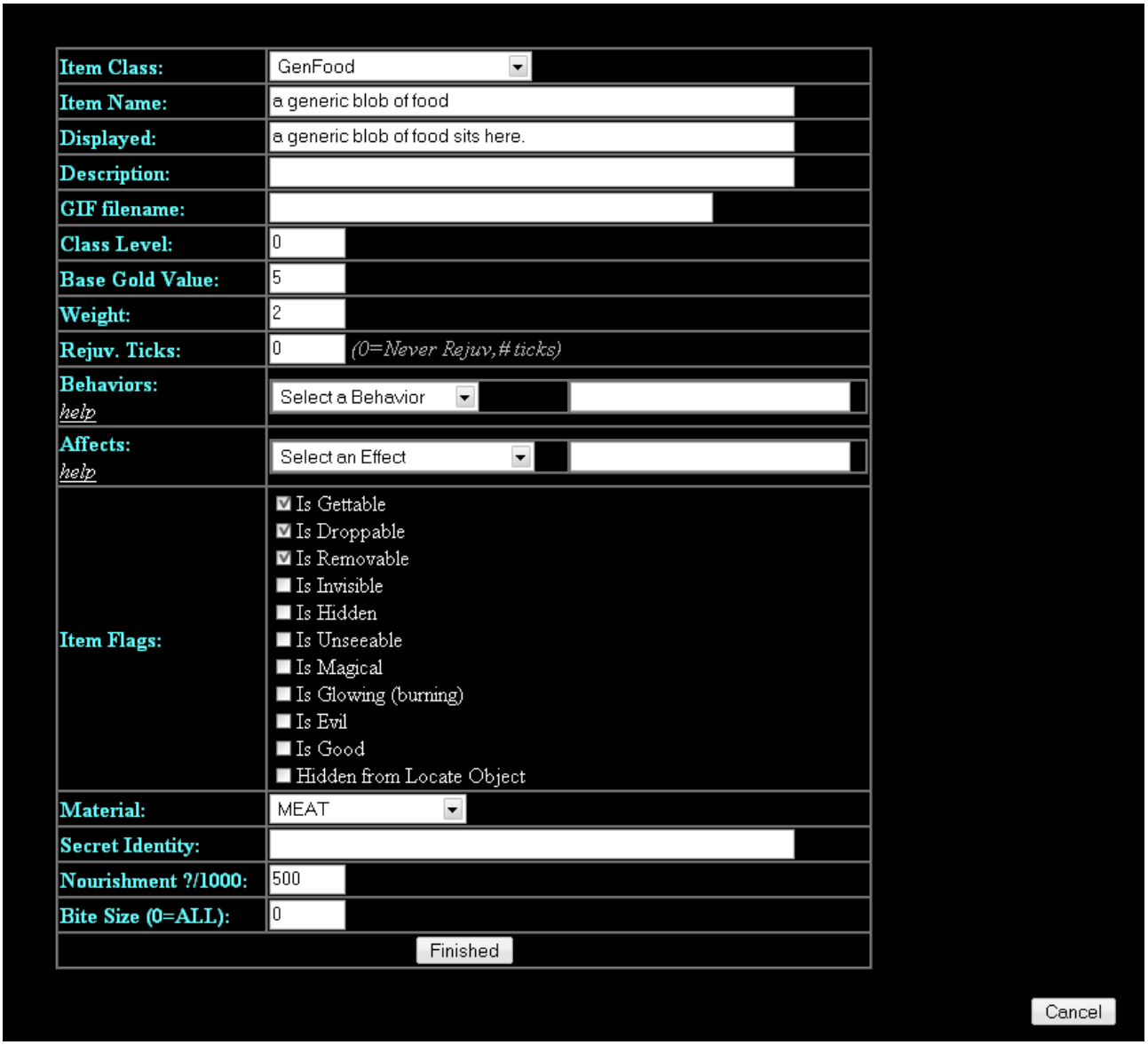

Figura 3.4 Editor de Objetos

Herramientas administrativas de MUD Grinder

Las pantallas que se utilizan para realizar funciones administrativas son las siguientes:

Menú Principal de MUD Grinder

Después del acceso, el administrador MUD ofrece un completo menú de opciones. (ver figura 3.5), que incluyen: Gestión de Memoria de Recursos, Visualizador de Jugador, Visualizador de Boletines, Gestor/Editor de Búsqueda, pantalla de Informe de Estado del Sistema, pantalla de Seguimiento del Sistema, Servidor de Sucesos MUD y Visualizador de Estadísticas de Uso MUD, (ver figura 3.7).

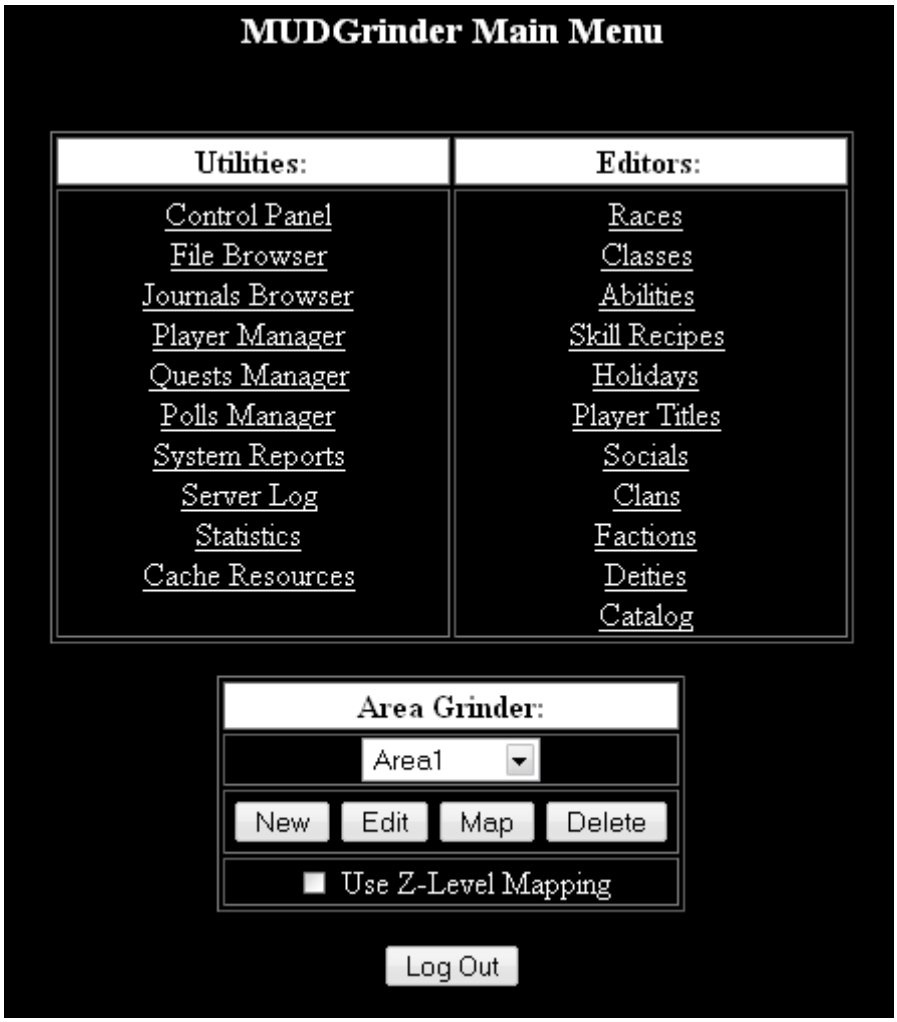

Figura 3.5 Menú Principal de MUD Grinder

Visualizador de Jugador

Una lista completa de todos los jugadores puede ser extraída, junto con un resumen de sus estados. Pinchando sobre un nombre se obtendrá una pantalla con información detallada del jugador, junto con algunas opciones de gestión, (ver figura 3.6).

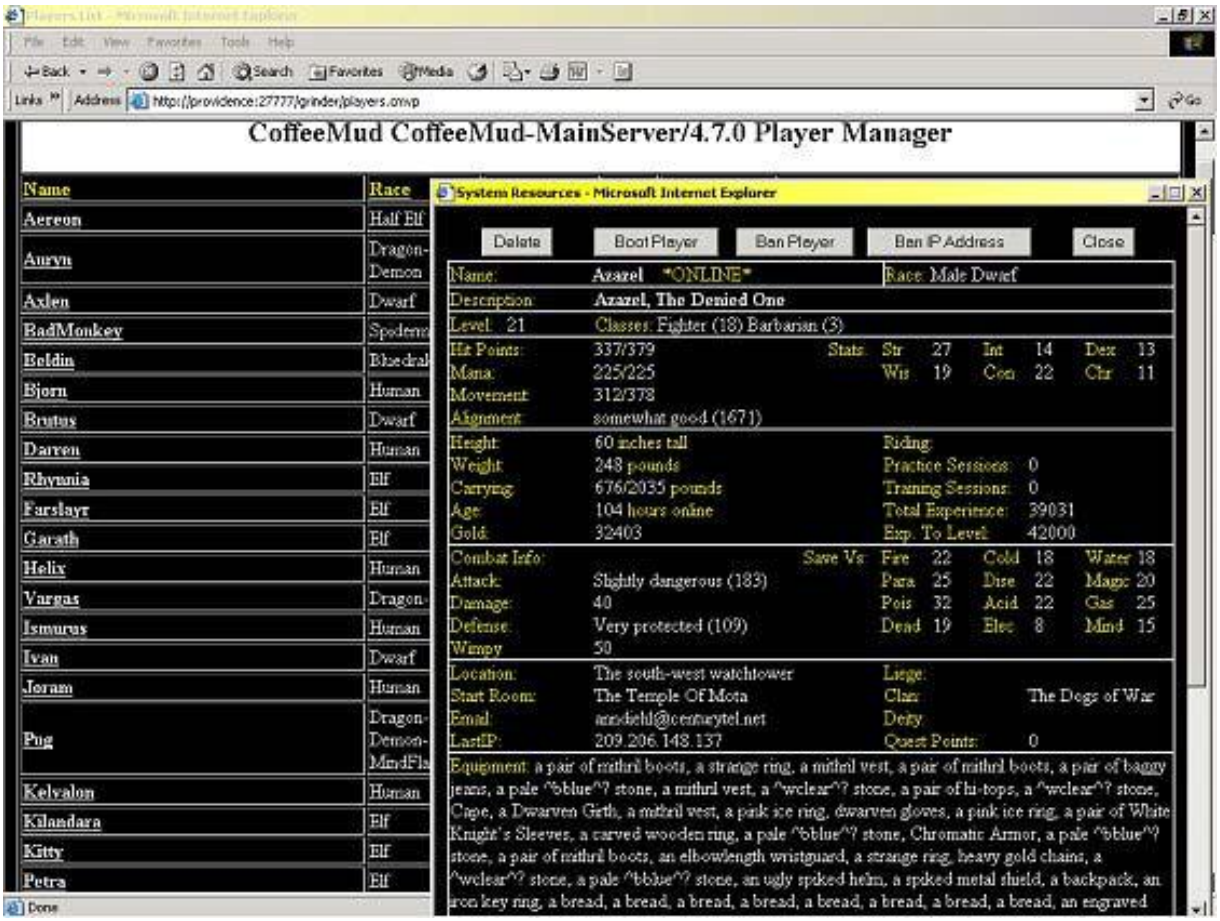

Figura 3.6 Visualizador de Jugador

Visualizador de Estadísticas

El Visualizador de Estadísticas Coffee MUD, (ver figura 3.7), guarda un completo juego de información estadística, para ser usado por el MUD. Se pueden visualizar estas estadísticas cada día, semana, mes o año, y agruparlas según se desee. También se pueden filtrar las estadísticas por clase de jugador, raza, nivel, género y otras categorías.

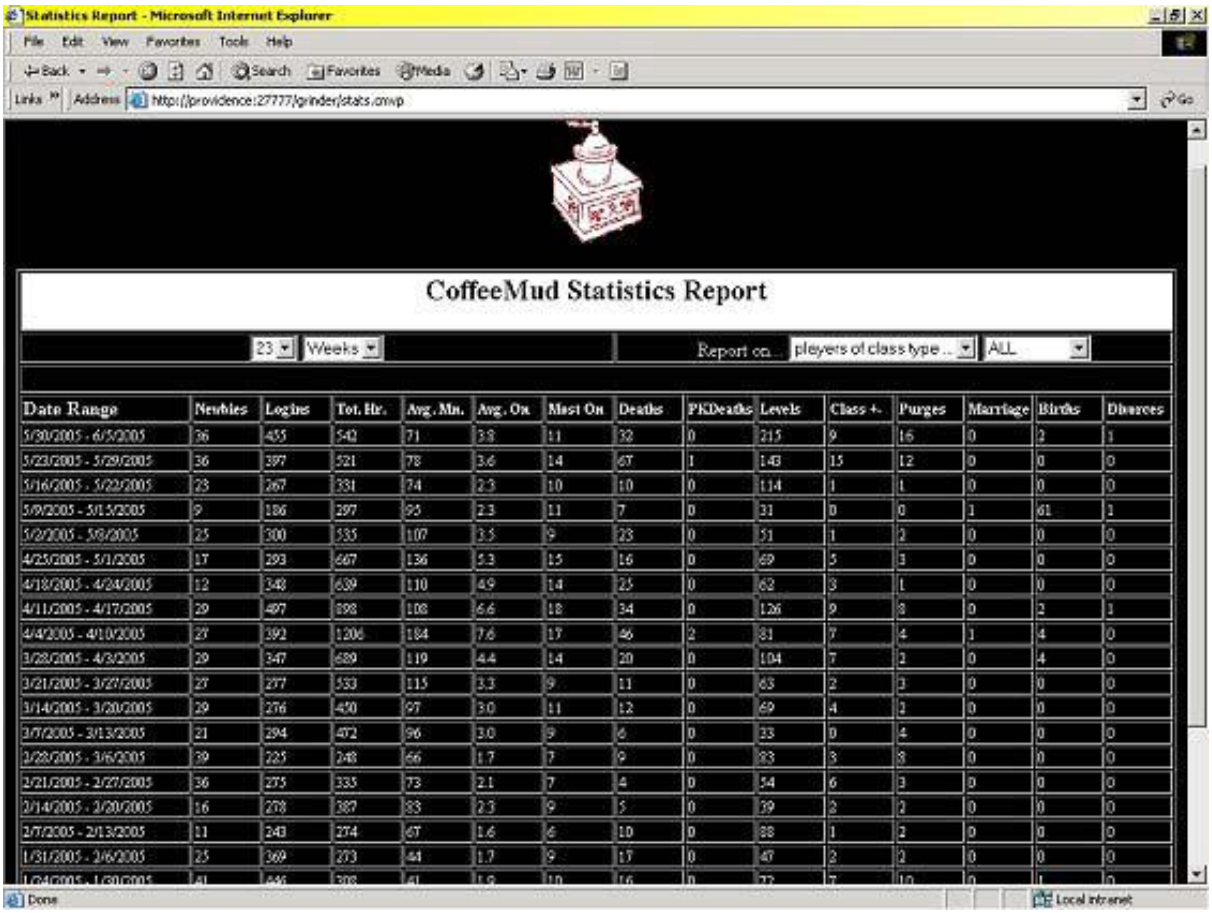

Figura 3.7 Visualizador de Estadísticas

3.5 Aplicación al proyecto

A continuación se describen de forma resumida los procesos de creación y edición que se han utilizado en este proyecto.

Proceso de creación de las áreas

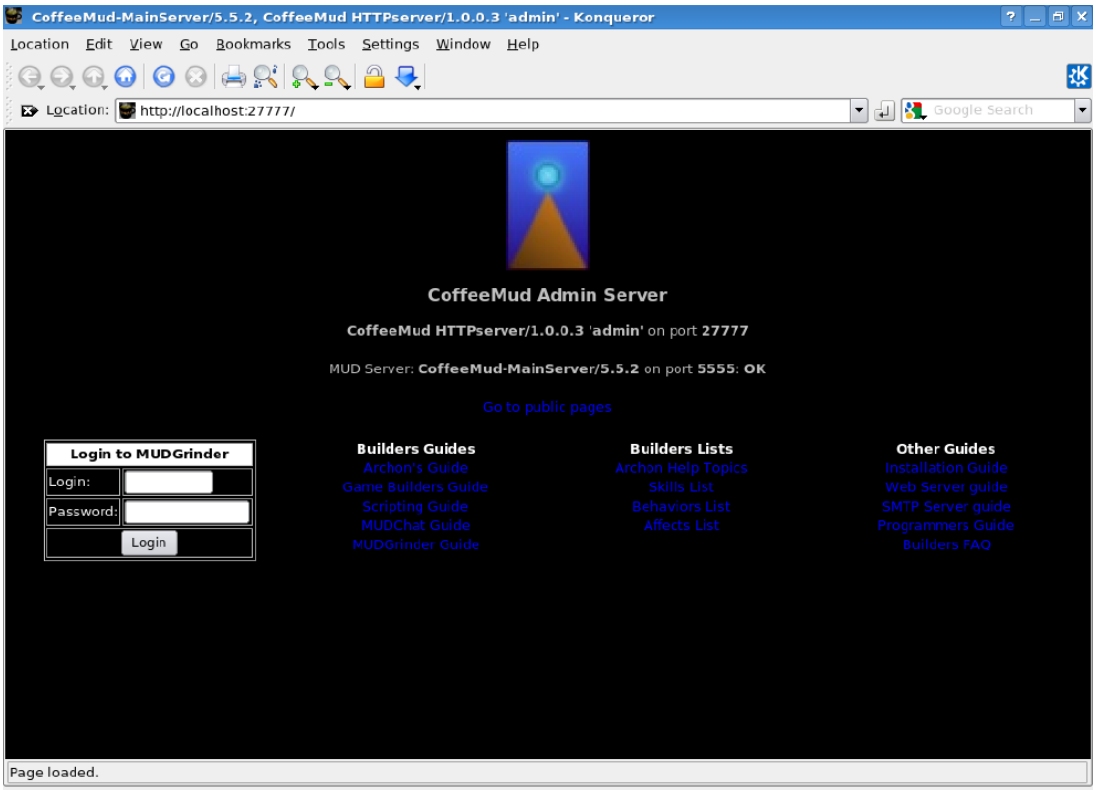

Figura 3.8 Pantalla de creación de área

Pantalla inicial de creación de las áreas. En ella se inicia el proceso introduciendo la identidad del usuario y la contraseña.

Pantalla de edición de las áreas

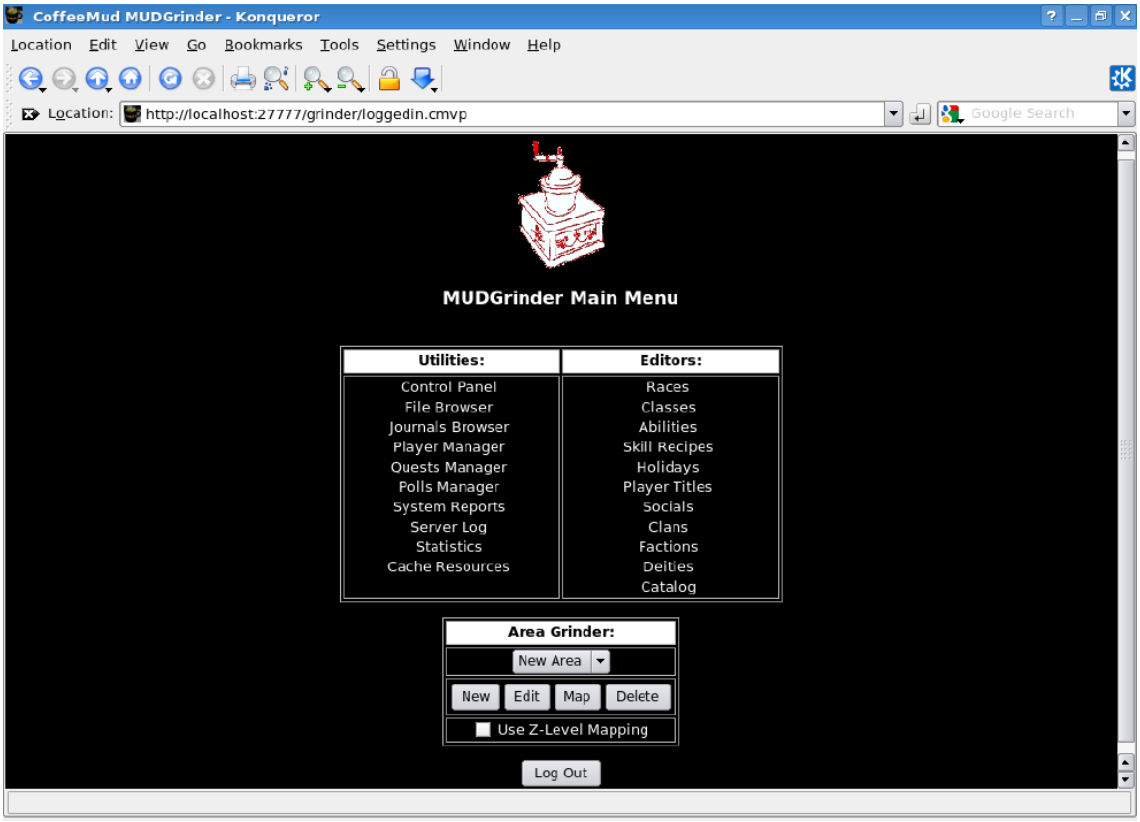

Figura 3.9 Pantalla de edición de áreas

Pantalla donde se crean y editan las áreas por donde se mueven los agentes, (Passage y Castle),. Para ello se pincha en el botón “New”. Para modificarle se utiliza el botón “Edit” y para determinar su ubicación de las diversas dependencias o habitaciones que constituyen el área en cuestión se utiliza el botón “Map”. El botón”Delete” se utiliza para borrar.

También permite modificar todas las opciones del juego, como la ubicación de los elementos que motivan a los agentes, (comida, bebida, medicina,..), así como las características y habilidades de los mismos, si procede.

Pantalla de creación del nombre del área

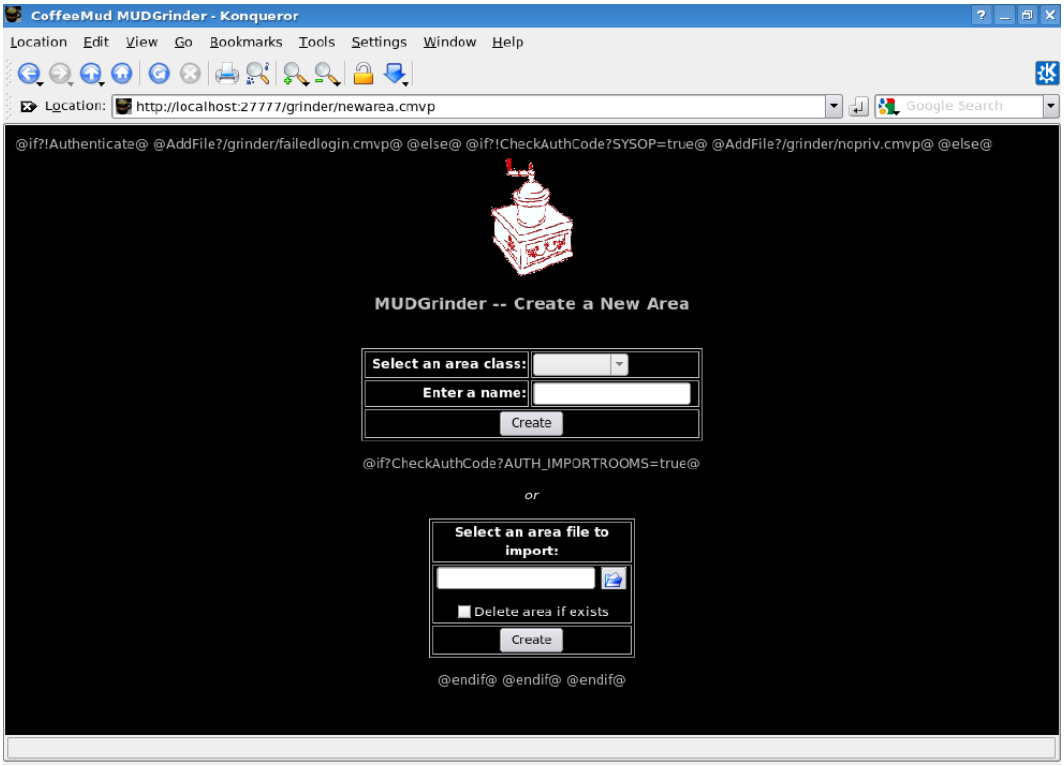

Figura 3.10 Pantalla de creación del nombre del área

El nombre del área se crea introduciéndolo en el campo “Enter a name”, (por ejemplo Passage o Castle). Se confirma pinchando en el botón “Create”.

Pantalla de creación de dependencias (habitaciones)

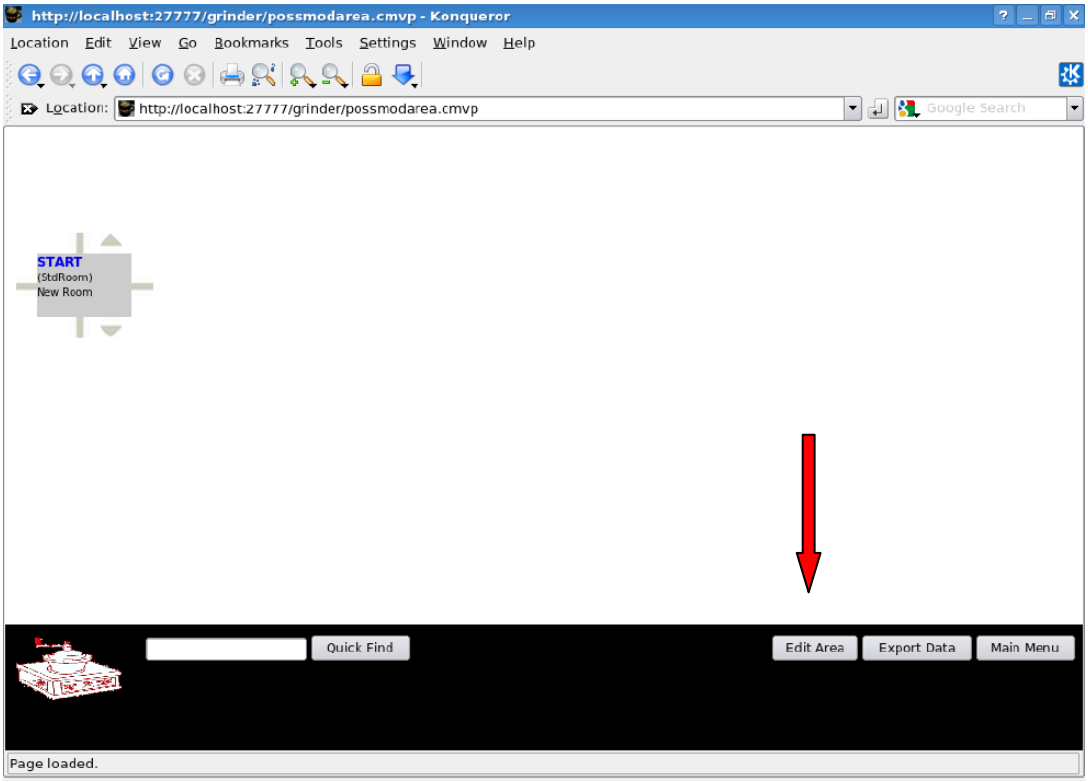

Figura 3.11 Pantalla de creación de dependencias

Esta pantalla aparece automáticamente cuando se presiona el botón “Create” de la pantalla anterior.

Sirve para crear la dependencia inicial del mundo virtual, a partir de la cual se van creando las sucesivas dependencias que constituyen el mapa de dicho mundo.

Pantalla de creación de nuevas habitaciones

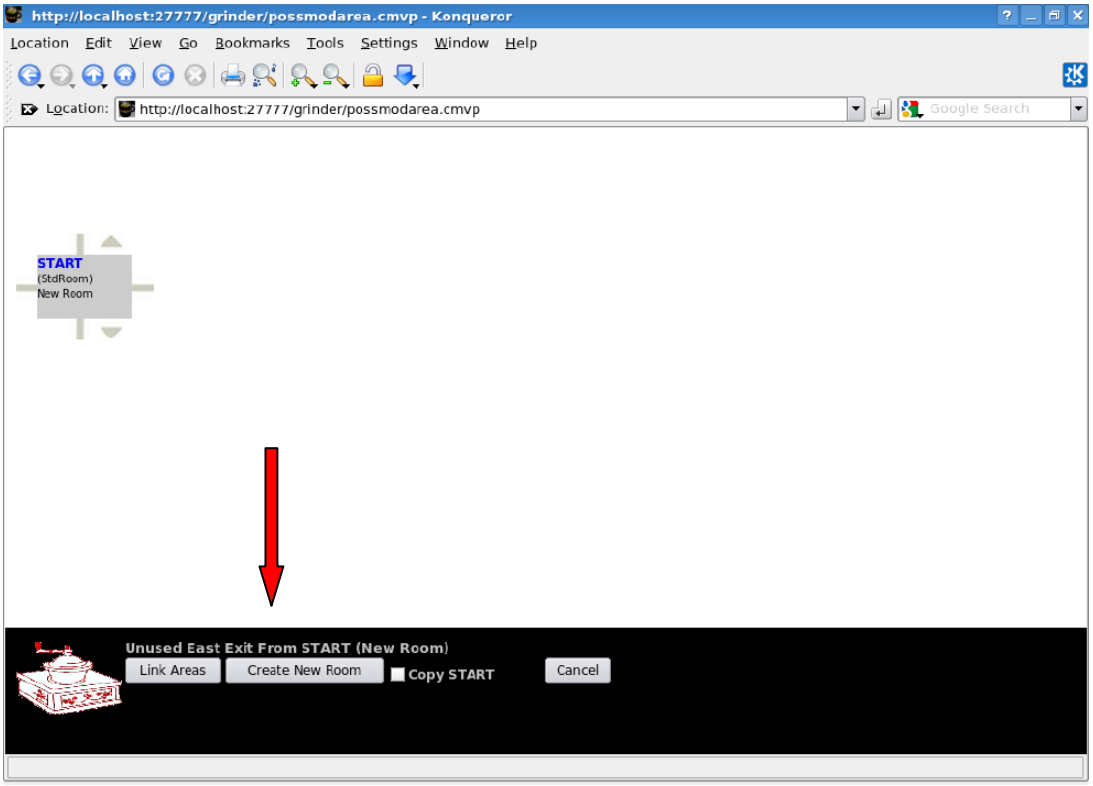

Figura 3.12 Pantalla de creación de nuevas habitaciones

Esta pantalla aparece cuando se selecciona una de las cuatro direcciones posibles de la pantalla de la figura anterior.
En la dirección seleccionada se crea una nueva dependencia unida a la ya existente.

Pantalla de edición de nuevas dependencias

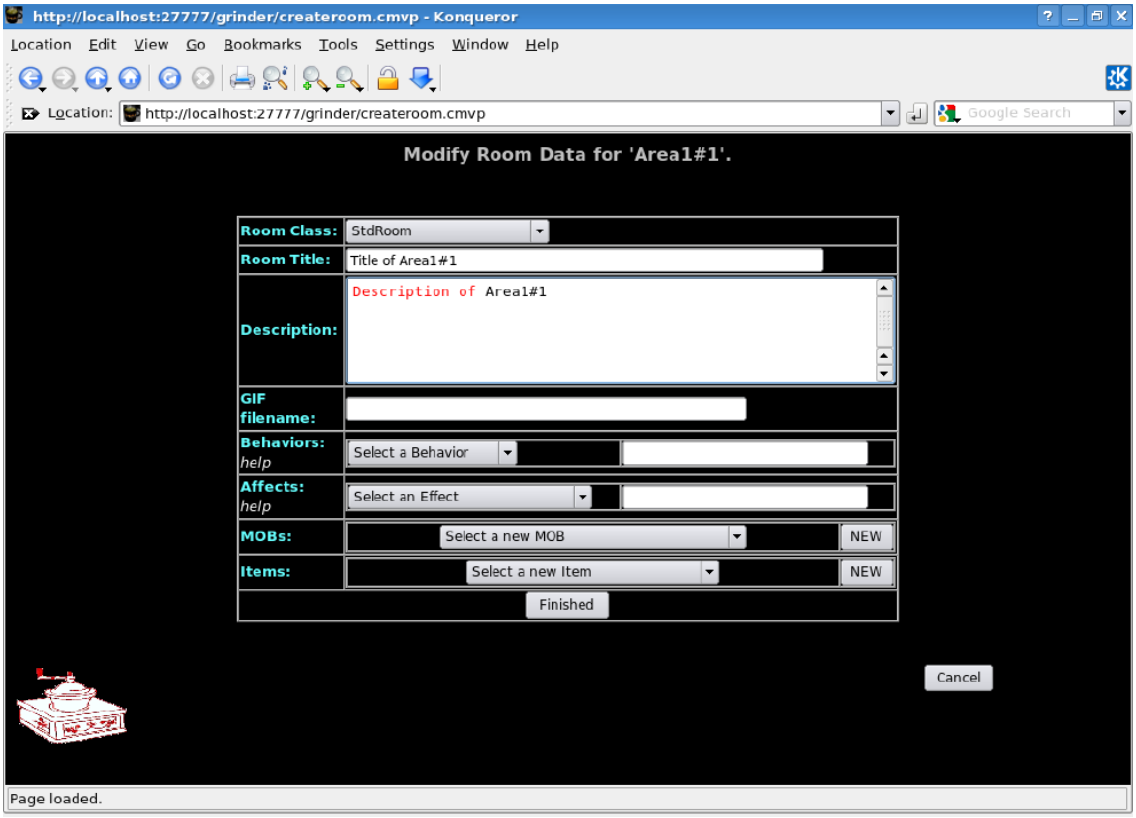

Figura 3.13 Pantalla de edición de nuevas dependencias

Mediante esta pantalla se crea el tipo, (pasillo, habitación, etc.) y características, (si tiene puerta o no, si está abierta o cerrada, etc.), de las nuevas dependencias, así como su contenido, (comida, bebida, medicina, etc.).

Pantalla con el resultado de añadir nueva habitación

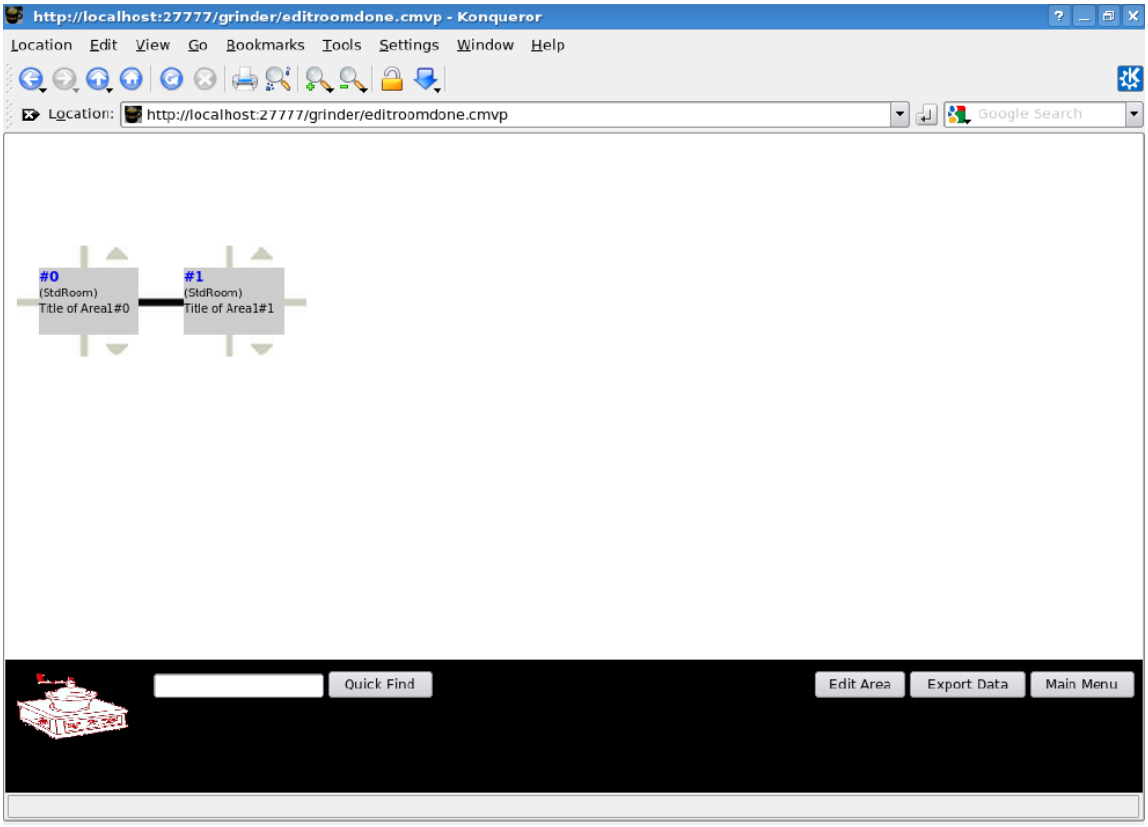

Figura 3.14 Pantalla resultado de añadir nueva habitación

En esta pantalla se muestra el resultado del proceso de creación de la figura 3.13.

La nueva dependencia aparece vinculada a la anterior en la dirección seleccionada de las cuatro posibles.

Área Castle

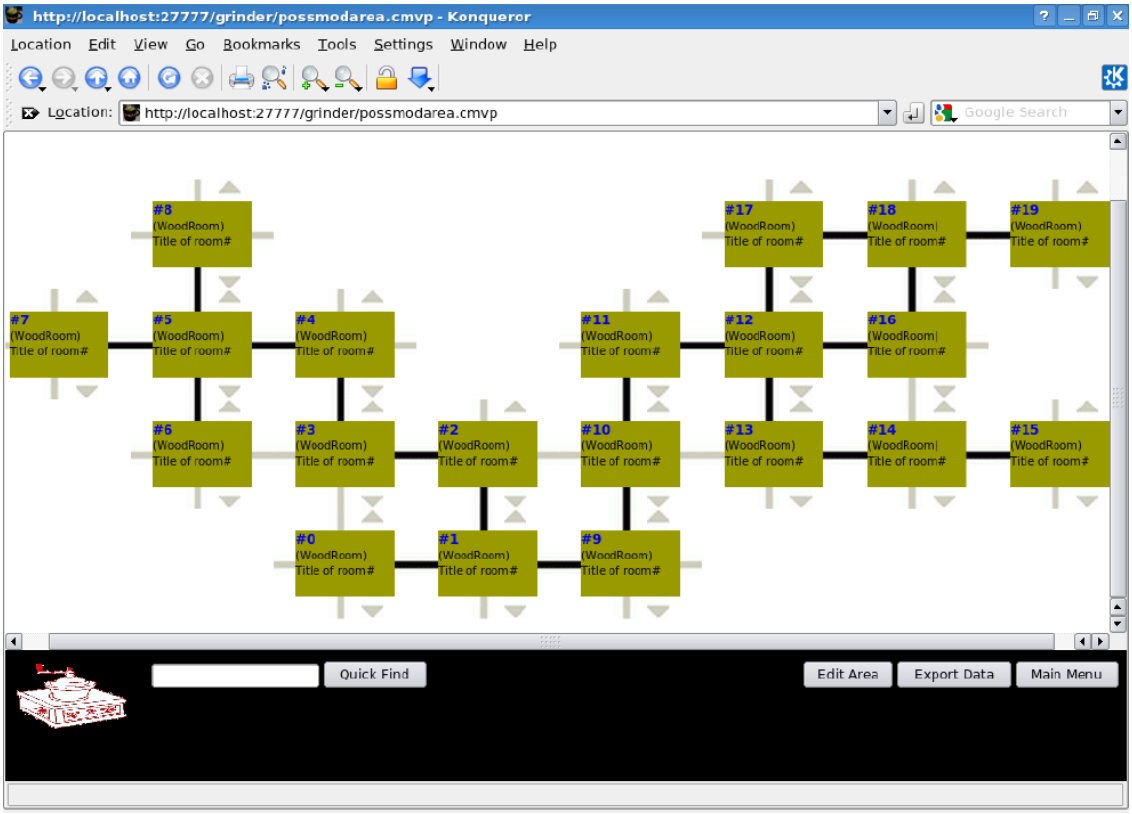

Figura 3.15 Área Castle

En la figura aparece la configuración final del área denominada “Castle”.
Esta área está constituida por veinte habitaciones según la disposición que aparece en la figura.

Área Passage

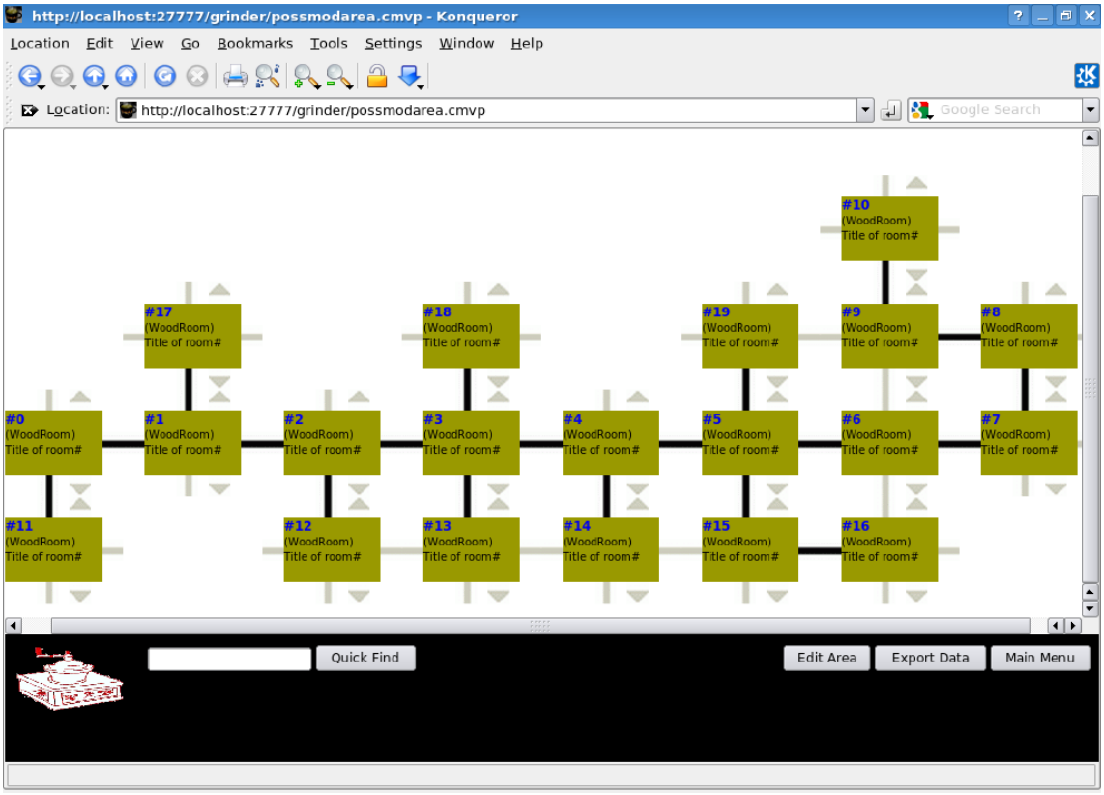

Figura 3.16 Área Passage

En la figura 3.16 se puede ver la configuración final del área denominada “Passage”.
El área ”Passage” está constituida a su vez por veinte dependencias dispuestas en torno a un pasillo central.

3.6 Descripción del Mundo virtual

En base a la utilización del Coffee Mud y con la ayuda de las herramientas descritas previamente, se ha creado el mundo virtual en el que se desarrolla el juego, base de este proyecto.

Se ha elegido el área denominada "Passage" para esta descripción. Dicho mundo virtual está constituido por un pasillo con doce dependencias centrales y ocho habitaciones situadas a ambos lados del pasillo.

En este entorno es donde se mueve el robot o agente quien se va encontrando con objetos, (estáticos), así como con otros jugadores, (dinámicos), con los cuales va interactuando lo que le permite aprender así como modificar su estado de bienestar.

Al principio del juego el agente no dispone de información sobre la ubicación de los objetos ni de los otros agentes externos. Cuando se mueve es cuando va memorizando la situación de los mismos.

El agente, dependiendo de su estado, realiza una serie de acciones como comer, beber, etc. Asimismo se relaciona con otros individuos con los que se encuentra en el mundo virtual como forma de adquirir experiencia.

En las distintas dependencias que configuran el mundo virtual el agente puede encontrarse con objetos, como comida, bebida o medicinas que le permiten cubrir sus necesidades primarias. También puede encontrarse con otros agentes lo que le permite relacionarse socialmente y aprender pautas de comportamiento al respecto.
Los objetos estáticos permanecen siempre en la posición inicial, de manera que existen habitaciones con comida, otras con agua y otras con medicinas. En cambio los otros agentes se van moviendo de forma autónoma.

El "Passage" es un entorno abierto, es decir no existen puertas, por lo que el agente para moverse sólo necesita desplazarse en una determinada dirección, (norte, sur, este y oeste). Los comandos utilizados para interactuar con los objetos activos y estáticos serán descritos posteriormente, ya que forman el conjunto de acciones disponibles para cada agente.
Entre estas acciones es interesante destacar las de "ir a" y "explorar":

- "Ir a", utiliza un algoritmo Dijkstra para elegir el camino más corto entre la habitación donde se encuentra el agente y la habitación a donde se propone llegar.

- "Explorar", utiliza un algoritmo DFS, (Deep First Search), que determina una ruta de exploración del entorno desde la dependencia donde se encuentra el agente.

3.7 Interfaz Gráfica del entorno

Como se ha mencionado en el punto anterior la relación entre el agente y el juego o interacción con el Mundo Virtual está basada en texto, por ello es muy difícil tener una visión de cómo progresa el juego ya que cada jugador sólo ve el recinto donde se encuentra. Para tener una visión global de todas las dependencias y del resto de los jugadores que configuran el juego, es necesario tener una visión del total de dependencias y objetos pasivos y agentes que se encuentran en el "Passage".

Esto se consigue mediante una interfaz gráfica que permite observar el conjunto de dependencias y agentes, lo que ofrece la visión de totalidad requerida. De esta forma se obtiene una ventana como la de la figura 3.17 da la visión completa de la situación del juego para cada jugador.

Nota.- Esta interfaz no ha sido desarrollada en este proyecto. Se incluye a título informativo.

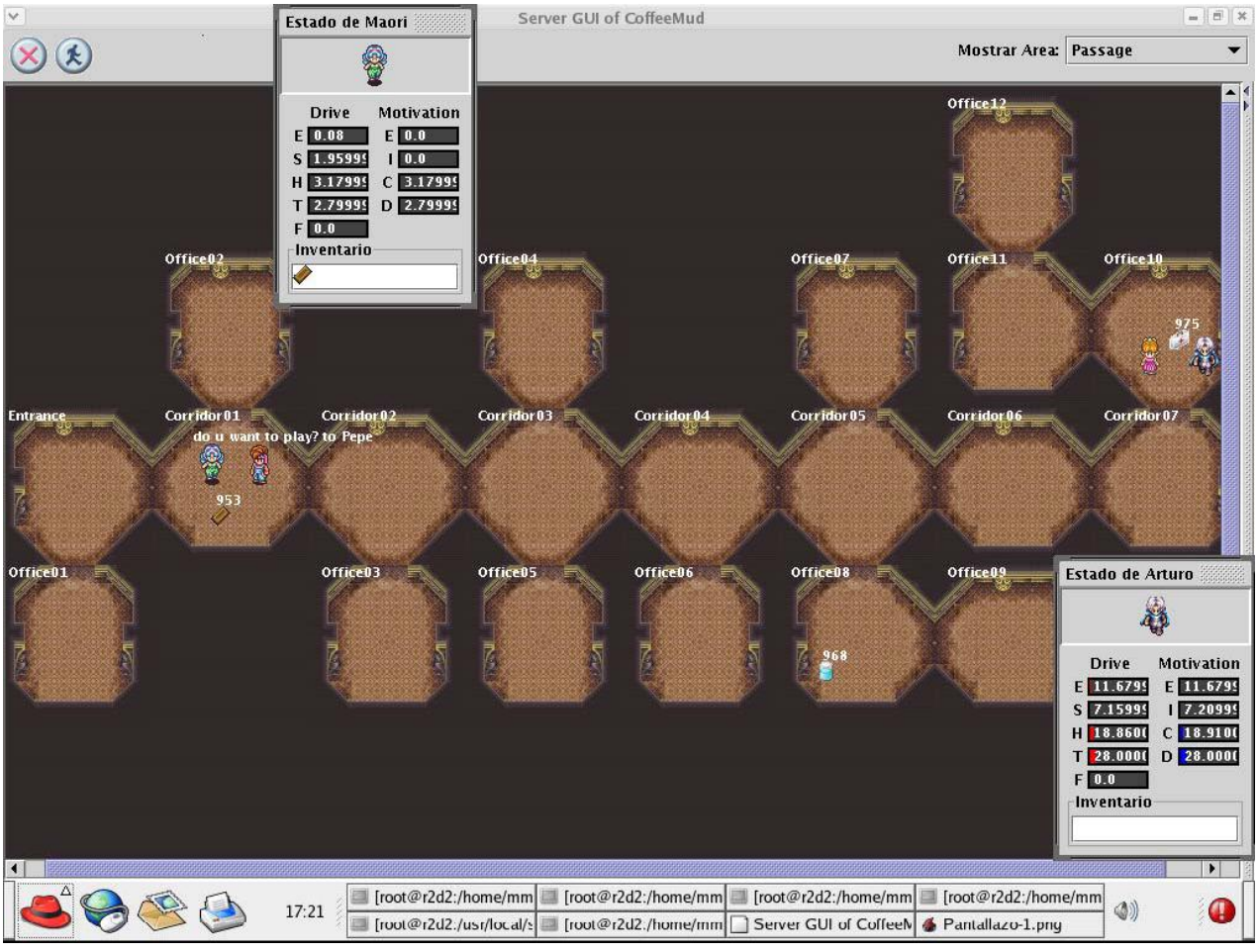

Figura 3.17 Ventana de situación del juego

Esta interfaz gráfica permite seguir todas las acciones de los jugadores.

También permite ver los valores de los parámetros que definen la situación del jugador en cada instante del juego, como son;

- Drives: Energía (Energy), Social(Social), Salud (Health), Sed (Thirst), y Miedo (Fear).
- Motivaciones: Comer (Eat), Socializase (Interact), Tomar medicina (Cure), y Beber (Drink).
- Objetos que tiene en su poder (Inventario).

Todo ello gracias a una sub-ventana (ver Figura 3.18) para cada jugador.

Figura 3.18 Subventana de jugador

Capítulo 4:
BASE DE DATOS

4 BASE DE DATOS

Uno de los elementos clave en la plataforma experimental, objeto de este proyecto, lo constituye el sistema de almacenamiento de los datos que se van produciendo a lo largo cada partida o juego.

Los datos se almacenan en una base de datos para su posterior utilización, tanto en el análisis como en la representación gráfica de los resultados.

La Base de Datos es el equivalente a la memoria del agente, (como estoy, como he estado y que he hecho). Estos datos son utilizados para incrementar la experiencia del agente, modificando su estado como forma de aprendizaje.

Dichos datos son recuperados posteriormente mediante una "Interfaz Gráfica", realizada mediante la herramienta Qt que permite la visualización de los mismos a través de una representación en forma gráfica.

4.1 Visión general

Definición de Base de Datos

Una base de datos es una estructura lógica que sirve para almacenar un conjunto de información que es administrada y a la que se accede a través de un DBMS (Data Base Management System).

El modelo más difundido es el conocido como modelo relacional, llamado así por el uso de vínculos o enlaces entre los datos, estableciendo relaciones que facilitan la recuperación de la información. El modelo de bases de datos relacionales se basa en el establecimiento de relaciones naturales, lógicas, entre las distintas informaciones almacenadas. Se basan en la teoría de conjuntos habitual en el álgebra matemática, siendo SQL, (Structured Query Language), el lenguaje utilizado para componer las expresiones que operan sobre dichos conjuntos.

Las bases de datos relacionales organizan la información en tablas. Las tablas son semejantes a hojas de cálculo donde las filas corresponden a las instancias, y las columnas a los dominios. Cada tabla es un conjunto de datos tabulados, en forma de matriz bidimensional formada por una serie de filas y columnas. El cruce de cada fila y columna contiene un valor, una unidad atómica de datos que se ajusta a una determinada categoría de información.

Un DBMS relacional manipula la información relativa a los diferentes tipos de cosas del mundo real (entidades) en tablas que representan esas entidades. Una tabla es como una hoja de cálculo; cada renglón representa una entidad en particular (instancia), y cada columna representa la información respecto de la entidad (dominio). En ocasiones las entidades están hechas de entidades más pequeñas, como órdenes y líneas de orden.

Una base de datos relacional bien manejada, minimiza la necesidad de las aplicaciones de contener información respecto al almacenamiento físico de los datos que se van a acceder. Para maximizar el aislamiento de los programas de las estructuras de datos, los DBMSs

restringen el acceso a los datos mediante el protocolo SQL, un lenguaje no procedural que limita al programador a obtener ciertos resultados.

El lenguaje más común para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje de Consulta Estructurado, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

SQL fue comercializado por primera vez por IBM en 1981, el cual fue presentado a ANSI y desde entonces ha sido considerado como estándar para las bases de datos relacionales. Desde 1986 ha aparecido en diferentes versiones como: SQL:92, SQL:99 SQL:2003.

Tablas y sus relaciones

Una base de datos relacional es una base de datos basada en un modelo relacional. Estrictamente hablando el término se refiere a una colección específica de datos pero a menudo es usado como sinónimo del software usado para gestionar esa colección de datos. Ese software se conoce como sistema gestor de base de datos relacional o RDBMS (relational database management system).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario casual de la base de datos. La información puede ser recuperada o almacenada por medio de «consultas» que ofrecen una amplia flexibilidad y poder para administrar la información.

Tablas en las bases de datos, se refiere al tipo de modelado de datos, en que se guarda en el los datos recolectados por un programa. Su estructura general se asemeja a la vista general de un programa de Hoja de cálculo.

Las tablas se componen de dos estructuras:

- **Campo**: Corresponde al nombre de la columna. Debe ser único y además de tener un tipo de dato asociado.
- **Registro**: Corresponde a cada fila que compone la tabla. Allí se componen los datos y los registros. Eventualmente pueden ser nulos en sus almacenamientos.

En la definición de cada campo, debe existir un nombre único, con su tipo de dato correspondiente. Esto es útil a la hora de manejar varios campos en la tabla, ya que cada nombre de campo debe ser distinto entre sí.

A los campos se les puede asignar, además, propiedades especiales que afectan a los registros insertados. El campo puede ser definido como índice o autoincrementable, lo cual permite que los datos de ese campo cambien solos o sean el principal indicador a la hora de ordenar los datos contenidos.

Cada tabla creada debe tener un nombre único en la cada Base de Datos, haciéndola accesible mediante su nombre o su sinónimo (dependiendo del tipo de base de datos elegida).
La estructura de las tablas viene dado por la forma de un archivo plano, los cuales en un inicio se componían de un modo similar.

Todas las filas de una tabla son idénticas en estructura pero distintas en contenido. Esto significa que cada fila tendrá el mismo número de columnas, con el mismo nombre y tipo de información, pero el valor contenido en cada una será diferente. Las columnas son también conocidas como campos y como atributos.

Cada una de las columnas de una tabla se asocia a un dato indivisible, a veces se dice que es un elemento *atómico* para destacar esa naturaleza, como puede ser el nombre de una persona o el importe de una transacción.

Una fila es el conjunto de columnas que corresponden a un mismo objeto, por ejemplo los datos de una persona o una transacción. Cada fila tiene siempre la misma estructura pero el contenido será distinto.

El conjunto de tablas que forma la base de datos ha de tener una coherencia, es decir, las tablas tendrán algún tipo de relación entre sí, a pesar de que los datos que contengan sean diferentes.

La relación entre dos tablas de una base de datos se encuentra en la coincidencia entre los valores existentes en unas columnas concretas.

Acceso a los datos

Una de las reglas características de las bases de datos relacionales, indica que cada valor existente en una base da datos debe ser accesible mediante una combinación de tres elementos: nombre de la tabla a que pertenece, valor de la clave primaria que identifica la fila y nombre de la columna donde se encuentra. Surgen así dos de los atributos con que cuente toda tabla de una base de datos relacional: el nombre y la clave primaria.

Cada una de las tablas existentes en una base de datos al igual que cada columna de una tabla, debe contar con un nombre único. Esto significa que no deben existir dos tablas con el mismo nombre en la misma base de datos, ni dos columnas con el mismo nombre en la misma tabla.
Se denomina clave primaria a un atributo de cada tabla que indica el nombre de la columna cuyo valor identificará de manera única a cada fila existente.

En cierta forma la clave primaria es como el nombre de cada fila en una tabla, de manera que combinada con el nombre de ésta y de la columna permite crear referencias únicas para obtener o alterar un valor determinado.

Integridad de los datos

Otra de las reglas que caracterizan una base de datos relacional es la que indica que la integridad de la información debe ser mantenida en la propia base de datos, no en las aplicaciones que la utilicen.

El objeto Consulta

Este objeto es el recomendado para ejecutar las consultas. Es una subclase de *strstream* en cual se puede escribir como en cualquier otro *stream* para ayudar en la formación de la consulta.

Tipos de datos

En la tabla siguiente se enumeran los tipos de datos utilizados con mayor frecuencia, usando como nombre del tipo una denominación general.

Nombre	**Información que puede contener**
Bolean	Sí o no, verdadero o falso. Únicamente uno de dos valores posibles.
Char	Una secuencia de caracteres de longitud fija
Varchar	Una secuencia de caracteres de longitud variable.
Int	Un número entero sin parte decimal
Money	Un número relativo a importes económicos
Float	Un número en punto flotante, con parte decimal.
Date	Una fecha
Binary	Cualquier secuencia de bytes

Tabla 4.1 Tipos de datos

Conjunto de Resultados Estáticos

El resultado de una consulta también puede ser almacenado estáticamente en la que se llama una estructura SQL especializada. Estas estructuras son almacenadas en algunos contenedores STL tales como un vector o lista, o igual a un conjunto o multi-conjunto. Este tipo de conjunto asume que el programador conoce el resultado a mostrar.

Recuperabilidad

La recuperabilidad significa que, si se da algún error en los datos, hay un fallo de programa ó de hardware, el DBA (Administrador de base de datos) puede traer de vuelta la base de datos al tiempo y estado en que se encontraba en estado consistente antes de que el daño se causara.

Las actividades de recuperación incluyen el hacer respaldos de la base de datos y almacenar esos respaldos de manera que se minimice el riesgo de daño ó pérdida de los mismos, tales como hacer diversas copias en medios de almacenamiento removibles y almacenarlos fuera del área en antelación a un desastre anticipado. La recuperación es una de las tareas más importantes de los DBA's.

Integridad

La integridad de una base de datos significa que, la base de datos ó los programas que generaron su contenido, incorporen métodos que aseguren que el contenido de los datos del sistema no se rompan así como las reglas del negocio. Por ejemplo, un distribuidor puede tener una regla la cual permita que solo los clientes individuales puedan solicitar órdenes; a su vez cada orden identifique a uno y solo un proveedor.

Seguridad

Seguridad significa la capacidad de los usuarios para acceder y cambiar los datos de acuerdo a las políticas del negocio, así como, las decisiones de los encargados. Al igual que otros metadatos, una DBMS relacional maneja la seguridad en forma de tablas. Estas tablas son las "llaves del reino" por lo cual se deben proteger de posibles intrusos.

Disponibilidad

La disponibilidad significa que los usuarios autorizados tengan acceso a los datos cuando lo necesiten para atender a las necesidades del negocio.

Desempeño

El desempeño significa que la base de datos no cause tiempos de respuesta poco razonables. En sistemas muy complejos cliente/servidor y de tres capas, la base de datos es sólo uno de los elementos que determinan la experiencia de los usuarios en línea y los programas desatendidos. El desempeño es una de las mayores motivaciones de los DBA para coordinarse con los especialistas de otras áreas del sistema fuera de las líneas burocráticas tradicionales.

Control de la concurrencia.

En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

Almacenamiento Físico

El almacenamiento físico depende del tamaño de las páginas para tablas e índices, así como del formato utilizado para guardar los datos físicamente en el disco.
Los discos duros actuales tienen un tiempo de acceso menor de 10ms, lo que significa que se pueden hacer teóricamente 100 búsquedas por segundo.

4.2 Aplicación al Proyecto

Como se menciona previamente en este documento, uno de los desarrollos más importantes de este proyecto ha sido el orientado a crear una Base de Datos donde se almacenan todos los datos relativos a cada partida o juego: tanto los resultados de las acciones que realiza el jugador como los cambios de estado que se producen y su influencia en el aprendizaje del mismo.

Dichos datos son almacenados para su posterior utilización así como para su representación visual mediante una adecuada interfaz gráfica que permita su análisis y evaluación de resultados.

MySQL

Para la aplicación a este proyecto se ha elegido MySQL como sistema de gestión de Base de Datos relacional.

MySQL es un sistema de administración relacional de bases de datos para múltiples usuarios que utiliza el lenguaje SQL estandarizado para el almacenamiento, actualización y acceso a información. Soporta varios lenguajes de programación: C, C++, Eiffel, Java, Perl, PHP, Python y TCL.

MySQL Server es la base de datos de código fuente abierto más usada del mundo desarrollado y proporcionado por MySQL AB. MySQL AB es una empresa cuyo negocioconsiste en proporcionar servicios en torno al servidor de bases de datos MySQL.

MySQL es software de fuente abierta. Fuente abierta significa que es posible para cualquier persona usarlo y modificarlo. Cualquier persona puede bajar el código fuente de MySQL y usarlo sin pagar. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades. MySQL usa el GPL (GNU General Public License) para definir que puede hacer y que no puede hacer con el software en diferentes situaciones. Si el usuario no se ajusta al GLP o requiere introducir código MySQL en aplicaciones comerciales, puede comprar una versión comercial licenciada.

Aunque se encuentra en desarrollo constante, el servidor MySQL fue desarrollado originalmente para manejar grandes bases de datos mucho más grandes que MySQL por lo que ofrece hoy un conjunto rico y útil de funciones. Su conectividad, velocidad, y seguridad hacen de MySQL un servidor bastante apropiado para acceder a bases de datos en Internet.

Poco a poco los elementos que faltan en MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

API MySQL++

MySQL++ es una interfaz (API) de la base de datos relacional MySQL para C++. Su propósito es hacerla funcionar mediante contenedores STL.

El objetivo de esta API es hacer mucho más fácil el trabajo para realizar consultas con otras estructuras de la STL1. Para trabajar con esta API se sigue casi el mismo proceso que las demás API SQL.

- Se Abre una conexión.
- Se formulan y ejecutan las consultas.
- Iteración directa con el conjunto de resultados de las consultas realizadas.

Características y Funcionalidad

Algunas de las más importantes características del SGBD son:

- Escrito en C y C++.
- Trabaja bajo diferentes plataformas: AIX 4x 5x, Amiga, BSDI, Digital Unix 4x, FreeBSD 2x 3x 4x, HP-UX 10.20 11x, Linux 2x, Mac OS, NetBSD, Novell NetWare 6.0 , OpenBSD 2.5, OS/2, SCO OpenServer, SCO UnixWare 7.1.x, SGI Irix 6.x, Solaris 2.5, SunOS 4.x, Tru64 Unix y Windows 9x, Me, NT, 2000, XP, 2003.
- Desarrollo de APIs para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
- Procesos MultiHilo. Capacidad de trabajar servidores con varios procesadores.
- Provee sistema transaccional con la tabla Innodb.
- Velocidad cuando se manipula datos con el tipo de tabla Myisam.
- Velocidad en la utilización de joins y procesos de optimización.
- Soporta muchos tipos de columnas para las tablas: FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM y OpenGIS (Modelo Geométrico).
- Manejo de la memoria a través de manejo del buffer y caché .

Como conclusión, dadas sus características de capacidad de almacenamiento, facilidad de uso y disponibilidad de documentación, se ha considerado a MySQL como la herramienta idónea para este proyecto por capacidad, facilidad de uso y disponibilidad de documentación.

4.3 Almacenamiento de datos de Cliente (el jugador) en Base de Datos

En el caso de entornos multi-agente es necesario disponer de una serie de tablas que contengan los valores de las partidas realizadas así como del estado de los diversos agentes que interactúan en el juego.

Dichas tablas son descritas de forma detallada en este punto y pueden contener los datos siguientes:

- Datos del Jugador: Tabla Jugador.
- Datos de cada Partida: Tabla Partidas.
- Datos históricos de las Partidas realizadas: Tabla Datos.
- Datos de los valores Q(s,a) para cada par estado-acción: Tablas Q.

En las siguientes secciones se describen cada una de estas tablas.

Tabla Jugador

En ella se almacenan los datos del jugador como: nombre, lugar donde está situado, palabra clave, así como los parámetros que definen el perfil o personalidad del jugador en cuestión.

Dichos valores son:

VARIABLE	**DESCRIPCIÓN (Registro por jugador)**
ID_jugador int	Identidad del jugador, (P.e. Maori = 1)
Nombre (Nº caracteres: 20)	Nombre del jugador, (p.e. Maori)
Password (Nº caracteres: 20)	Password del jugador
Área (Nº caracteres: 20)	Lugar donde está, (p.e. = Passage)
Alpha 1 (flotante)	Parámetro que define la personalidad del jugador
Alpha 2 (flotante)	Parámetro que define la personalidad del jugador
Alpha 3 (flotante)	Parámetro que define la personalidad del jugador
Alpha 4 (flotante)	Parámetro que define la personalidad del jugador

Tabla 4.2 Tabla Jugador

Tabla Partidas

Esta tabla contiene los datos históricos de las partidas realizadas como: número de la partida, lugar y fecha de la misma.

Dichos datos son los siguientes:

VARIABLE	**DESCRIPCIÓN**
ID_partida int	Número de la Partida
Área (Nº caracteres: 20)	Lugar donde se realizó
Fecha	Fecha de la Partida

Tabla 4.3 Tabla Partidas

Tabla Datos

En esta tabla se almacenan los datos de la partida y los valores de las variables de estado del jugador:

VARIABLE	**DESCRIPCIÓN**
ID_partida int	Identidad de la Partida
ID_jugador int	Identidad del jugador
Tiempo int	Instantes de tiempo en que se divide la partida
Energy (flotante)	Variable indicativa del estado del jugador (resp. Energía)
Social (flotante)	Variable indicativa del estado del jugador (resp. social)
Health (flotante)	Variable indicativa del estado del jugador (resp. salud)
Thirst (flotante)	Variable indicativa del estado del jugador (resp. sed)
Motiv_eat (flotante)	Variable indicativa del estado del jugador (resp. Energía)
Motiv_cure (flotante)	Variable indicativa del estado del jugador (resp. salud)
Motiv_drink (flotante)	Variable indicativa del estado del jugador (resp.bebida)
Motiv_interact(flotante)	Variable indicativa del estado del jugador (resp. social)
Wb (flotante)	Bienestar del jugador

Tabla 4.4 Tabla Datos

Tablas Q

Como se ha descrito en el apartado 2, el algoritmo de aprendizaje por refuerzo Q–learning es el utilizado para este proyecto. A través de este algoritmo el agente aprende la política de comportamiento correcta mediante el refuerzo obtenido después de ejecutar una acción *a* en el estado *s*.

Para aprender estos valores el agente debe mantener m*n tablas donde m es el número de estados internos (número de motivaciones) y n es el número de objetos. De esta manera el agente debe aprender a qué hacer con cada objeto para cada motivación dominante, por ejemplo: qué hacer con la comida cuando tiene hambre, qué hacer con el agua cuando tiene sed, etc.

En el caso de este proyecto se mantendrán 20 tablas (5 motivaciones y 4 objetos) correspondientes a los siguientes estados:

	Comida	**Medicina**	**Agua**	**Otro Agente**
OK	Q_{no_se}	Q_{no_sc}	Q_{no_sd}	Q_{no_friend}
Energía	Q_{e_se}	Q_{e_sc}	Q_{e_sd}	Q_{e_friend}
Curarse	Q_{c_se}	Q_{c_sc}	Q_{c_sd}	Q_{c_friend}
Beber	Q_{d_se}	Q_{d_sc}	Q_{d_sd}	Q_{d_friend}
Interactúa	$Q_{interact\_se}$	$Q_{interact\_sc}$	$Q_{interact\_sd}$	$Q_{interact\_friend}$

Tabla 4.5 Tabla de Q

En estas tablas la nomenclatura utilizada es la siguiente: *Q motivación dominante_estado en relación al objeto*. Siendo:

- Motivaciones:
  - no: No hay motivación dominante.
  - e: Comer.
  - c: Curarse.
  - d: Beber.
  - interact: Socializarse.
- Estado de los objetos:
  - se: Objeto comida.
  - sc: Objeto medicina.
  - sd: Objeto agua.
  - friend: Otro jugador.

Las tablas relacionadas a los objetos, es decir, las tablas con subíndice *se*, *sc*, y *sd*, tienen dimensión igual a: *número de estados en relación a los objetos * número de acciones*. Por lo tanto, tal y como se presento en el capítulo 2, la dimensión será *8 (estados)*4 acciones*.

Mientras que las tablas Q relacionadas con la interacción social (Q_friend) tienen unas dimensiones de matrices de *7 (acciones de interacción)*7 (acciones de interacción).*

Cada tabla tiene los siguientes campos:

- ID_p int (Identidad de la Partida).

- ID_j int (Identidad del Jugador).

- Tiempo int (Instante de la partida).

- Y además los campos relativos a los valores Q.

Cliente

El cliente ha sido modificado para almacenar todas estas variables.

Lo primero que se hace es una función que se encargara de rellenar la tabla Partidas (Tabla 4.3). Se le pasa como datos el área donde va a jugar el agente y si va a ser una partida nueva o si se une a una partida ya iniciada por otro agente. Dicha función tomara la fecha del sistema y realizara una consulta a la base de datos para ver cuál es la última partida creada, de modo que si es el caso de una partida ya iniciada por otro agente devolverá ese valor, o lo incrementara una unidad si es una partida nueva, y posteriormente rellenará la tabla partidas.

Se guarda el valor de la partida en una variable para rellenar el resto de tablas.

A continuación se crean dos funciones más. Una rellenará la tabla Datos (Tabla 4.4) con los valores generados por el agente. La otra función se encargará de gestionar los datos de las tablas Q del algoritmo de aprendizaje Q – learning. Al tener una estructura matricial se usan dos bucles para recoger los pares de datos de acción *a* y estado *s*.

Como conclusión la Base de Datos constituye la "memoria" donde se almacena toda la información necesaria para analizar y el comportamiento de cada jugador y para posteriores desarrollos del proyecto.

Capítulo 5: INTERFAZ GRÁFICA DE REPRESENTACIÓN DE DATOS

5 INTERFAZ GRÁFICA DE REPRESENTACIÓN DE DATOS

5.1 General

Con objeto de disponer de una visión global de los datos que configuran el juego, se ha desarrollado una interfaz gráfica de comunicación e información al jugador. En dichas gráficas se representa la evolución ó vida del jugador así como la de su experiencia ó aprendizaje.

La observación de dichos datos de los jugadores y del desarrollo de las partidas permite analizar el comportamiento de los jugadores con objeto de sacar conclusiones y poder desarrollar mejoras futuras en su aprendizaje.

Dicha interfaz se ha diseñado basándose en la *herramienta Qt.*.

5.2 Herramienta Qt

Qt se puede definir como una herramienta software utilizada para la creación de Interfaces Gráficas de Usuario, (GUI), por la empresa Trolltech.

Consiste en un conjunto de librerías multiplataforma para el desarrollo de aplicaciones GUI escritas en C++, lo que indica que está orientada a objetos.

Qt permite crear aplicaciones para diversas plataformas, como son: Windows desde Windos 95 hasta XP, Mac OS X, Linux Solaris, HP-UX y otras versiones de Unix con X11. El desarrollo de una aplicación o GUI para una determinada plataforma no implica cambios importantes si se quiere utilizar en otra plataforma diferente.

Qt tiene un sistema de programación basado en un entorno de ventanas y dispone de una herramienta de trabajo denominada Qt designer. Los diseños creados sobre Qt son automáticamente traducidos a C++. El paquete Qt incluye las librerías Qt, las herramientas Qt designer y tmaka.exe, así como un directorio de programas ejemplo.

Para este proyecto se ha utilizado la distribución *Qt educational edition 3.3.3,* que incluye algunas herramientas útiles como son:

- Uic: User Interface Compiler.
- Moc: Meta object Compiler.
- Tmake: Herramienta para crear makefiles.
- Qt designer: Aplicación para diseñar ventanas de diálogo de forma gráfica.

Sus características más relevantes son:

- Dispone de una librería para interfaces gráficos.
- Es orientada a objetos. Usa el lenguaje C++.
- Se basa en el concepto de Widgets (objetos de una paleta), Señales, Slots y Eventos.
- Qt también dispone de otras funciones.
- Librerías básicas.

- Entrada/Salida, Manejo de Red, XML.
- Interfaces con Bases de Datos: MySQL, Oracle.
- Plugings (interfaces con Bases de Datos):

    - QODBC3 ODBC (Open Database Connectivity) Driver.
    - QOCI8 Oracle Call Interface Driver Oracle 8 and 9.
    - QPSQL7 PostgreSQL v6.x and v7.x Driver.
    - QTDS7 Sybase Adaptive Server and Microsoft SQL Server Driver.
    - QMYSQL3 MySQL Driver.
    - QDB2 IBM DB2.
    - QSLITE SQLite.
    - QIBASE Interbase.
    - Sólo están disponibles para la versión libre QMYSQL, QODBC y QPostgreSQL.
    - Las bases de datos libres que podemos encontrar son: Firebird, Mysql, PostgreSQL, SQLite y Sybase ASE Express Edition (para linux).

Proceso de diseño

El proceso seguido para diseñar la aplicación utilizada en este proyecto se puede resumir en los pasos siguientes:

- Creación del proyecto iniciando el Qt designer:
  Menu File/New/Open y se elige C++ project.

- Creación de la ventana principal de diálogo:
  Menu/File/New y después Main Window.

- Creación del archivo principal (main) del proyecto:
  Menu/File/New y C++ Main File, OK.

- Creación de la interfaz gráfica:

    - Creación del menú de la aplicación en la ventana principal: Se sitúa el puntero en un lugar vacío de la ventana y se pulsa el botón derecho del ratón. Aparece un menú que permite editar y dar nombre a los menús.

    - Añadir widgets (botones): Se realiza con el menú Tools, (herramientas). Los widgets creados se pueden posicionar con el menú Layaout/Grid.

    - Crear funciones: Se realiza con el menú Edit/Slots.

- Desarrollo de interfaces. De forma muy resumida los principales pasos son:

    - Entorno de desarrollo. Se abre en el menú de inicio con Desarrollo/Entorno de desarrollo.
    - Menú Proyecto/Nuevo Proyecto.

- Se selecciona C++/Qmake project/aplicación.
- Declaración del nombre y ruta del proyecto.
- Definición y asignación de nombre de Clase de Interface.
- Implementación de clase derivada.
- En menú Construir/Construir Proyecto y después Construir/Ejecutar programa principal.

5.3 Aplicación al Proyecto

La visualización de los datos que determinan el juego, al ser el reflejo de lo acontecido durante las partidas que constituyen la parte esencial del proceso de aprendizaje, es fundamental para sacar conclusiones y desarrollar posibles mejoras.

La interfaz gráfica permite observar el desarrollo de dichas partidas así como la evolución del estado del jugador y de los parámetros que determinan el proceso de aprendizaje de los jugadores o agentes.

Características de la interfaz gráfica:

- Mostrar los datos de referencia de cada partida: número de partida, número del agente, área en la que se desarrolla la partida, y fecha de creación.
- Elección de hasta 5 curvas a representar con 5 colores diferentes (pudiendo añadir más sólo con variar el código del programa).
- Selección entre 4 jugadores.
- Selección desde la partida 1 al valor máximo que tenga el tipo de variable "entero" en el sistema operativo donde se ejecute el programa.
- Selección de la variable a representar.
- Posibilidad de calcular el valor medio de los datos y el porcentaje de la zona de seguridad y mostrar dichos valores en la representación gráfica, además de añadir una recta señalizando la zona de seguridad.
- Posibilidad de añadir o borrar curvas dinámicamente. Pudiendo añadirlas o borrarlas sin que haya que cerrar la ventana de la representación.
- Redimensionamiento dinámico de las curvas al variar el tamaño de la ventana de la representación.
- Inclusión de 6 divisiones en ambos ejes con redondeo al alza del número de división.
- Si hay datos positivos y negativos se añade automáticamente una recta correspondiente al valor 0.
- Si una de las variables a representar es "bienestar" el máximo en el eje de abscisas corresponderá al valor 100.
- Posibilidad de guardar la representación en un archivo de imagen, en dos formatos.

A continuación se describen algunos ejemplos de utilización de las interfaces gráficas desarrolladas.

Ventanas

Como se ha mencionado anteriormente, la herramienta Qt permite la utilización de ventanas en el proceso de introducción de datos y la posterior visualización de resultados.
Al inicio de la partida aparecen dos ventanas iniciales:

- Ventana con los datos de la partida.
- Ventana con la elección de las variables a representar.

Ventana con datos de partida

La ventana con los datos de la partida contiene los datos siguientes: número de la partida, número del agente, fecha de creación de la partida, nombre del agente y lugar donde se está desarrollando, (Passage), ver Figura 5.1.

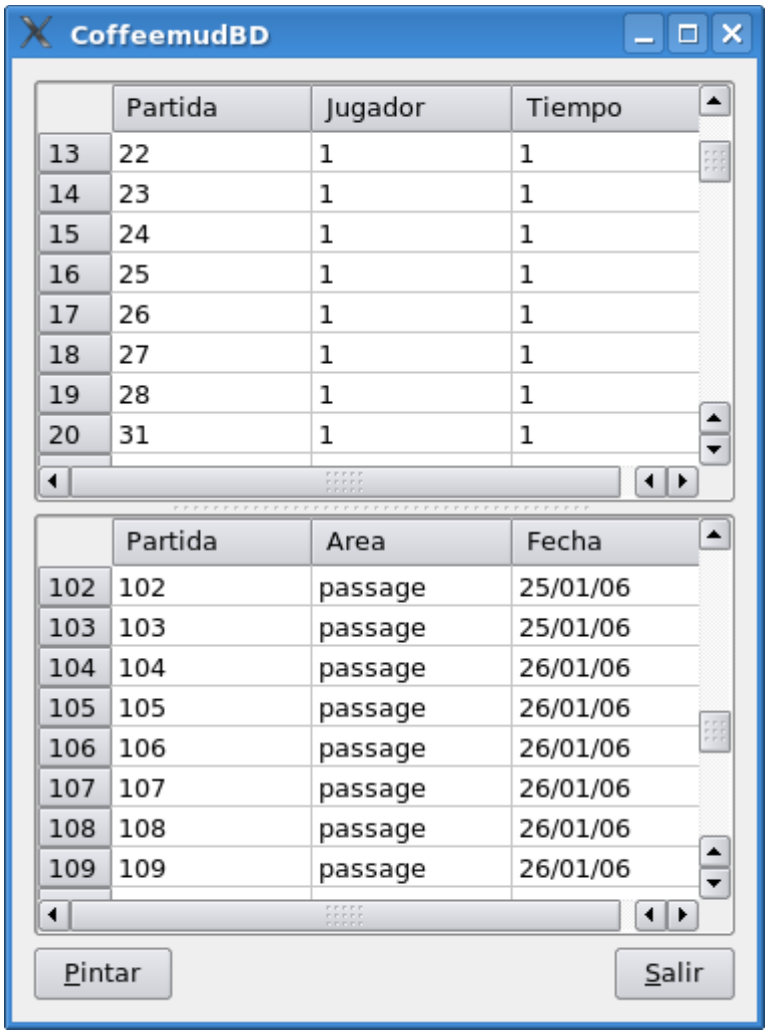

Figura 5.1 Ventana datos de partida

Ventana de elección de variables a representar

Si en la ventana anterior se elije la opción “Pintar”, aparece una nueva interfaz que permite elegir las variables a representar. Ver Figura 5.2.

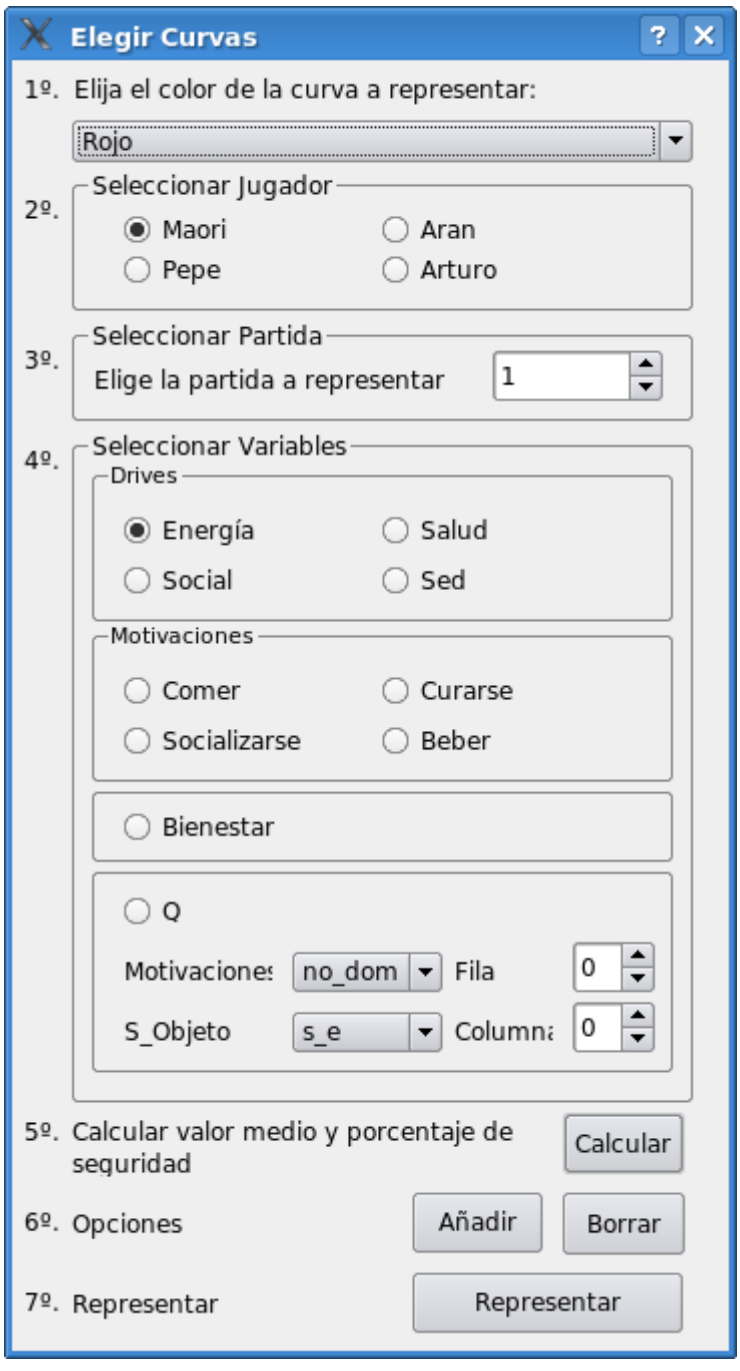

Figura 5.2 Ventana de elección de variables

Esta ventana permite la elección de las variables que definen la gráfica que se quiere visualizar. La ventana está dividida en 7 puntos, que corresponden con el orden que hay que seguir para representar los datos deseados:

1) Se comienza eligiendo el color de la gráfica en el comboBox situado debajo del letrero "Elija el color de la curva a representar:". Los colores que se han incluido son: rojo, verde, azul, cyan y magenta; se eligieron por ser colores que se diferencian fácilmente entre ellos.

2) En el punto 2 se elige el jugador por su nombre, (por ejemplo Maori) y se ve cuantos jugadores hay en la partida (de 1 a 4).

3) En tercer lugar el número de la partida a representar.

4) En el punto 4 aparecen las distintas variables de las que anteriormente se han recogido los datos para introducirlos en la base de datos.

   Las variables pueden ser las siguientes:

   - Drives.
   - Motivaciones.
   - Bienestar.
   - Los valores Q, por cada motivación dominante y objeto.

   Además, se puede calcular la zona de seguridad de la variable "Bienestar". Se considera que cuando el bienestar del agente está por encima de un valor determinado (92) está en la zona óptima: Zona de Seguridad. Si se presiona el botón de "Bienestar" se añade la zona de seguridad representada por una línea gris en el valor determinado (92) y se redimensiona la gráfica para los márgenes 0 y 100.

5) El punto 5 es una opción, que si se selecciona, en la leyenda de la gráfica se mostrará el valor medio y el porcentaje de seguridad de la variable elegida.

6) En este punto aparecen dos botones:

   a. Seleccionando el botón "Añadir" se guarda la selección.
   b. El botón "Borrar" se explicará más adelante.

7) Por último, se selecciona el botón de representar y aparece la representación de los datos o variables en una ventana nueva.

A continuación se dan algunos ejemplos de visualización de distintas variables seleccionadas.

Representación del *drive* Energía para el jugador "Maori", ver Figura, 5.3.

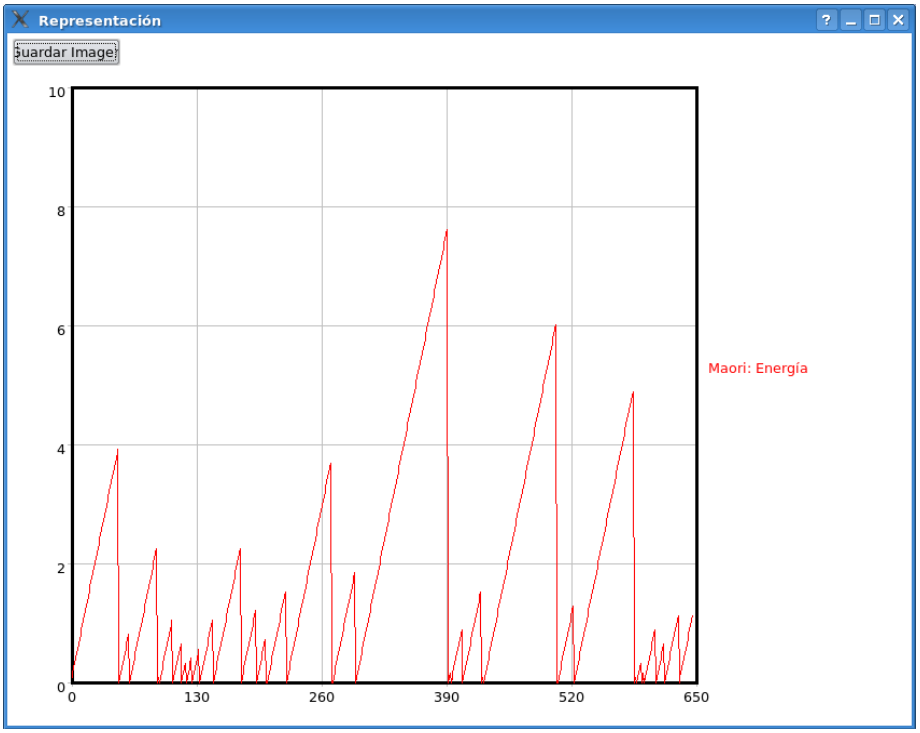

Figura 5.3 Gráfica de energía

Asimismo, mediante el campo 5 de la gráfica de la figura 5.2, puede visualizarse la evolución de los valores medios valores medios como se observa en la gráfica de la Figura 5.4.

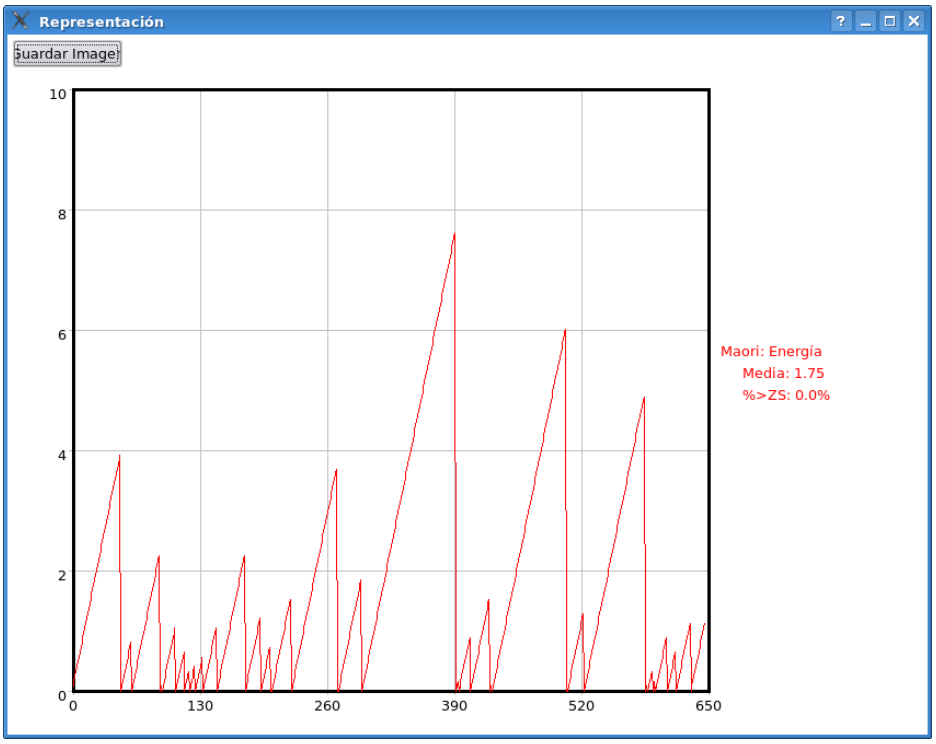

Figura 5.4 Gráfica de valores medios

Además, tal y como se ha explicado anteriormente, se pueden combinar en la misma gráfica distintas variables como: Energía, Salud, etc. Ver figura 5.5.

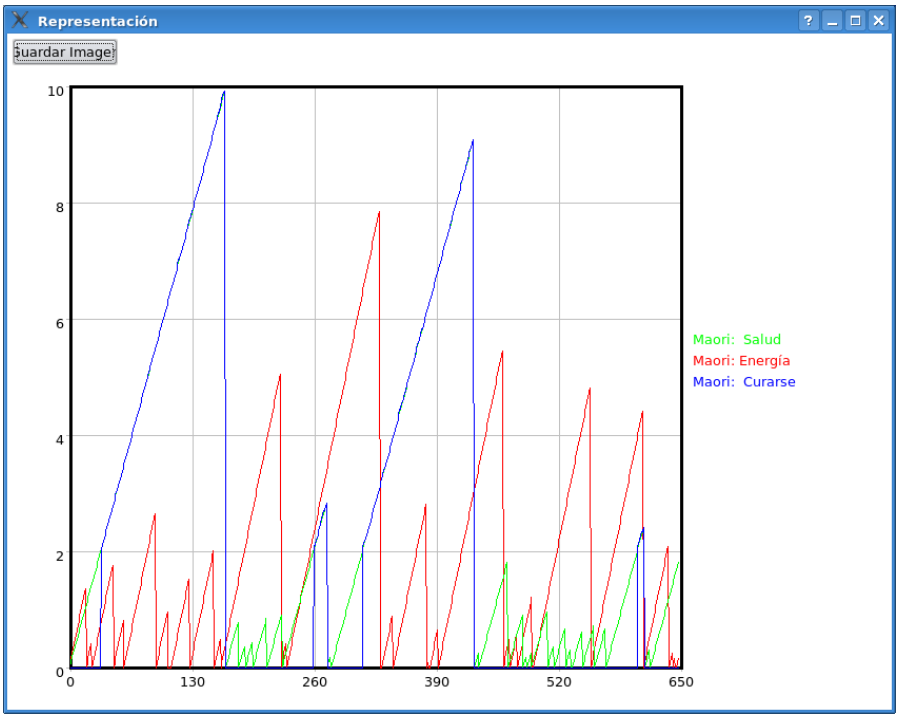

Figura 5.5 Gráfica de Energía, Salud, Curarse

También pueden visualizarse los valores Q(s,a). Ver como ejemplo las figuras 5.6 que sea añadido Qcomer y Qcurarse del agente Maori, y 5.7 donde se han añadido Qcomer de los agentes Pepe y Maori.

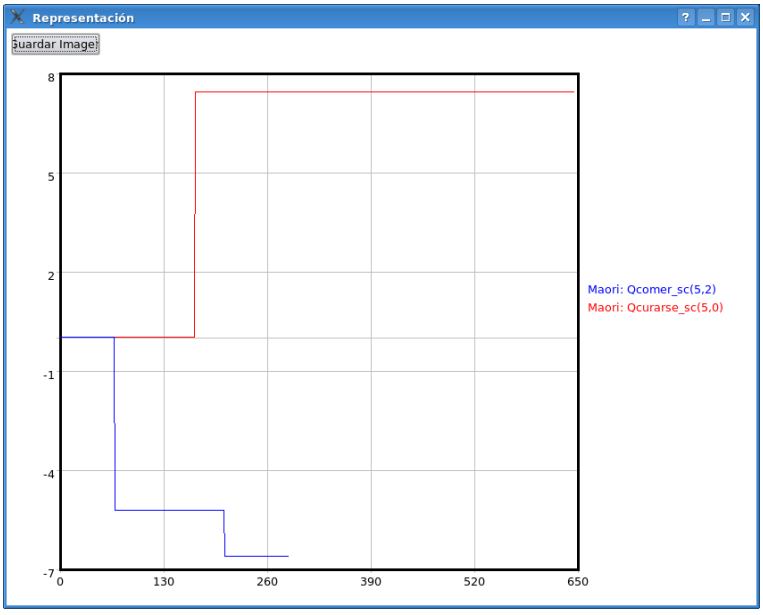

Figura 5.6 Gráfica de Q comer, Q curarse

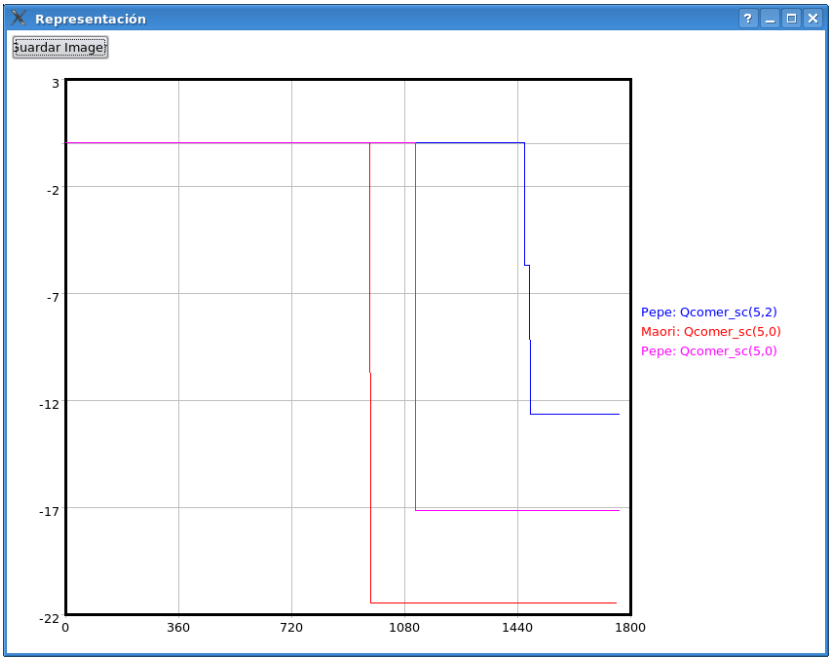

Figura 5.7 Gráfica de Q comer (Pepe, Maori)

Por otro lado, si se quiere comparar diferentes variables, se pueden mostrar hasta cinco representaciones distintas, cada una con su propio color y su propia leyenda. Ver figura 5.8, en la que se han añadido el bienestar de distintas partidas, además de las variables comer y curarse, todas son del agente Maori.

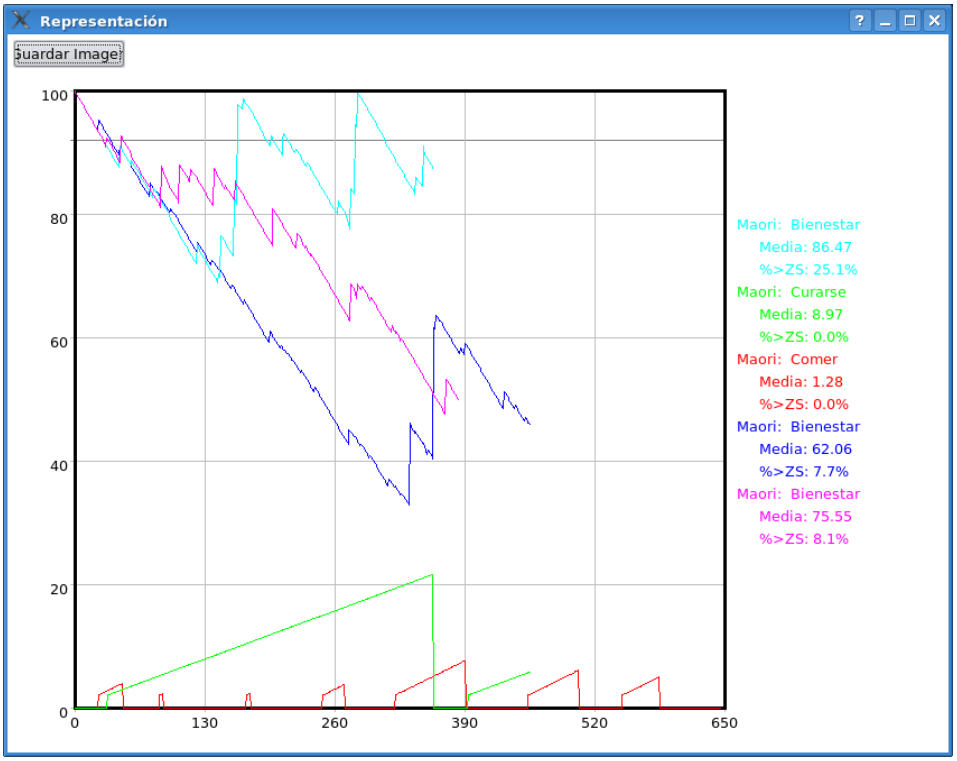

Figura 5.8 Gráfica de cinco variables

A continuación se realizan una serie de pruebas para ver cómo se comporta el programa. Se elige una partida con casi 1800 datos para comprobar que no hay problemas con el numero de datos seleccionados, se toma la variable Energía como referencia y se contrasta los datos de los agentes Maori, Pepe y Aran.

El programa ha sido diseñado para que haya 5 divisiones (6 contando los ejes) y las divisiones en el eje de abscisas y de ordenadas sean un número entero y no correspondan a números decimales. De manera que si el número más alto en el eje X es el 1736, el valor de la división será de 1800. Así la gráfica no queda sin cuadrar (Figura 5.9).

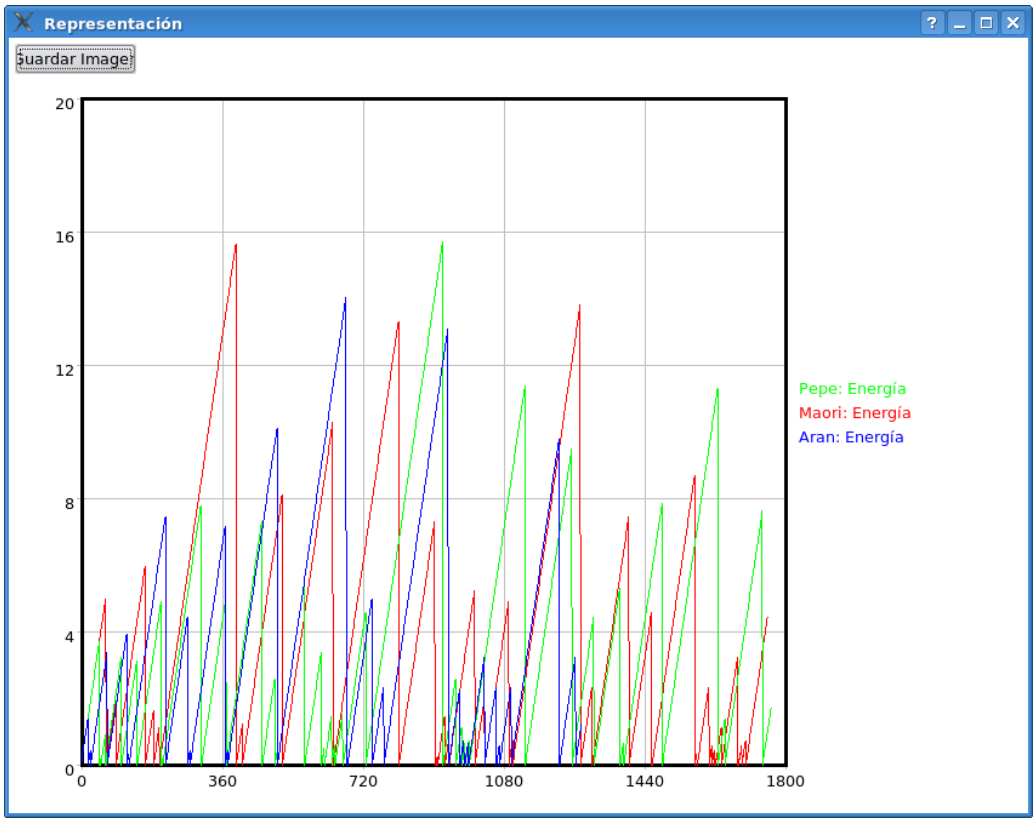

Figura 5.9 Gráfica de Energía (Pepe, Maori, Aran)

Otra de las características del programa es que se puede *eliminar* una de las variables representada sin borrar todas las que tenemos. Esto puede hacerse eligiendo el color de una curva que se quiere borrar y pulsando el botón “borrar” en el punto 6 de las opciones que aparecen en la ventana elegir curva (Figura 5.2). Si se selecciona el color azul y seleccionamos el botón “borrar” se pasa de la figura 5.9 a la figura 5.10.

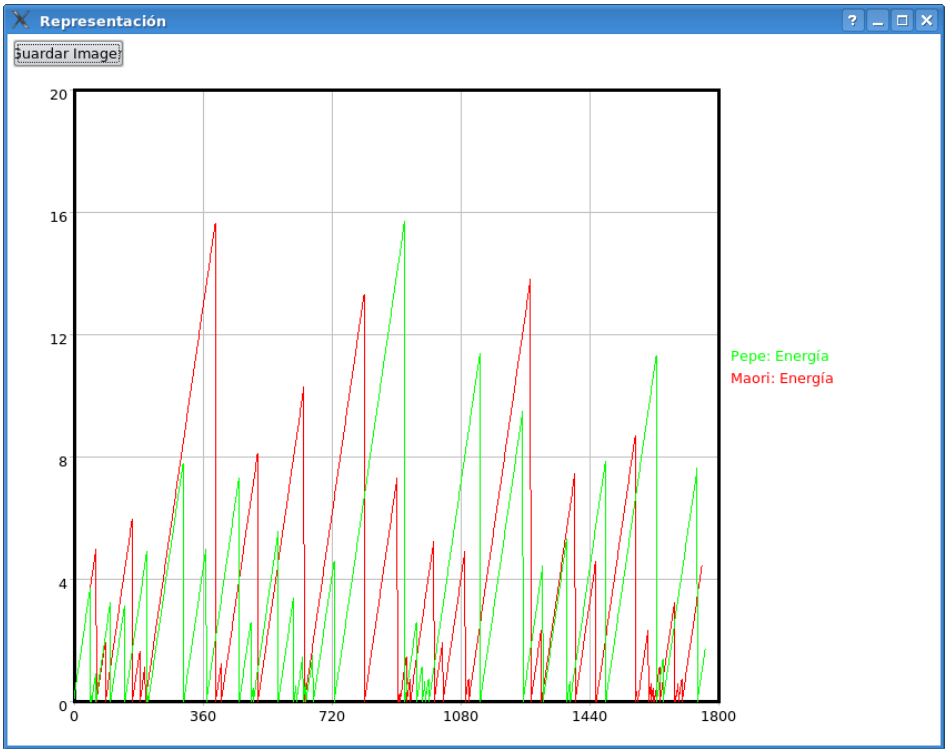

Figura 5.10 Gráfica de Energía (Pepe, Maori)

También se puede *añadir* más variables en la misma gráfica aunque no tenga el mismo número de datos a representar. El programa redimensionará automáticamente la gráfica ajustando los límites superior e inferior. Por ejemplo, de la figura 5.11 se puede pasar a la figura 5.12. En dicha gráfica se han añadido las variables energía y salud de distintas partidas.

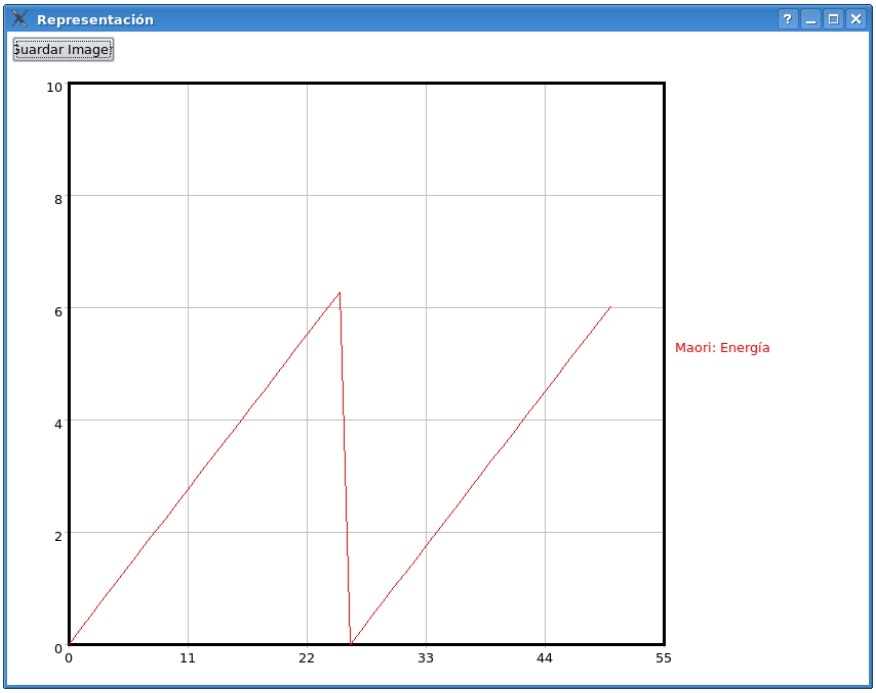

Figura 5.11 Gráfica de energía (Maori)

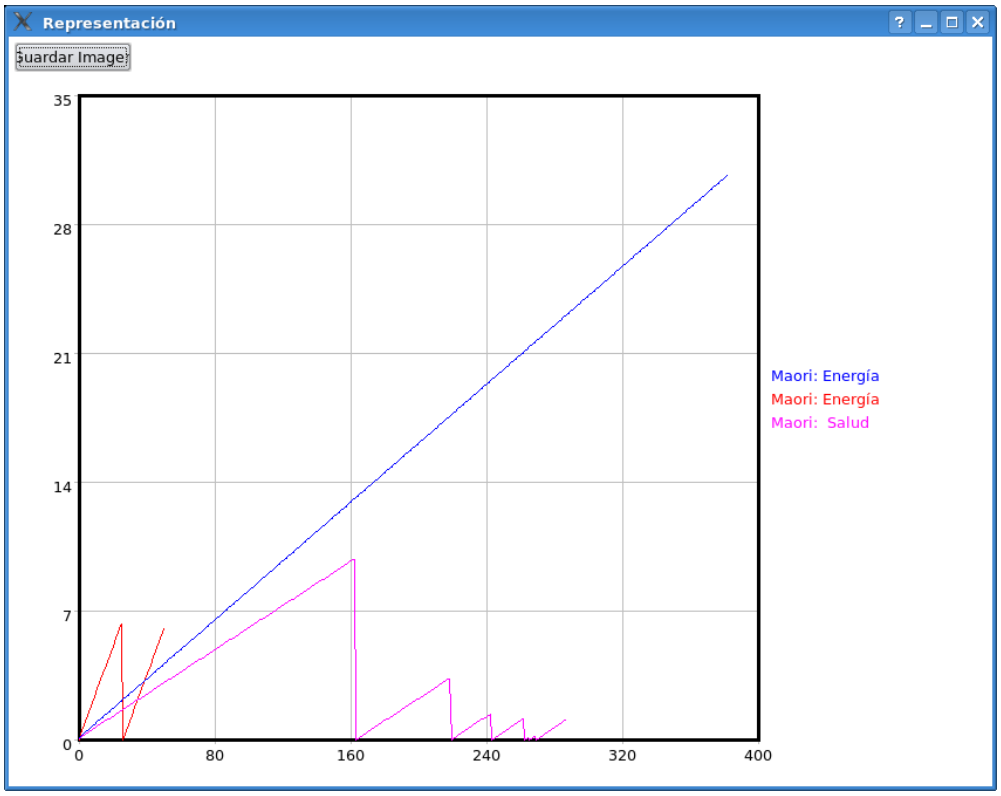

Figura 5.12 Gráfica de Energía, Salud (Maori)

Si la variable a añadir es “bienestar” el máximo corresponde a 100 y se añade una recta de color gris que corresponde a la zona de seguridad mencionada anteriormente se pasa de la Figura 5.13 a la Figura 5.14.

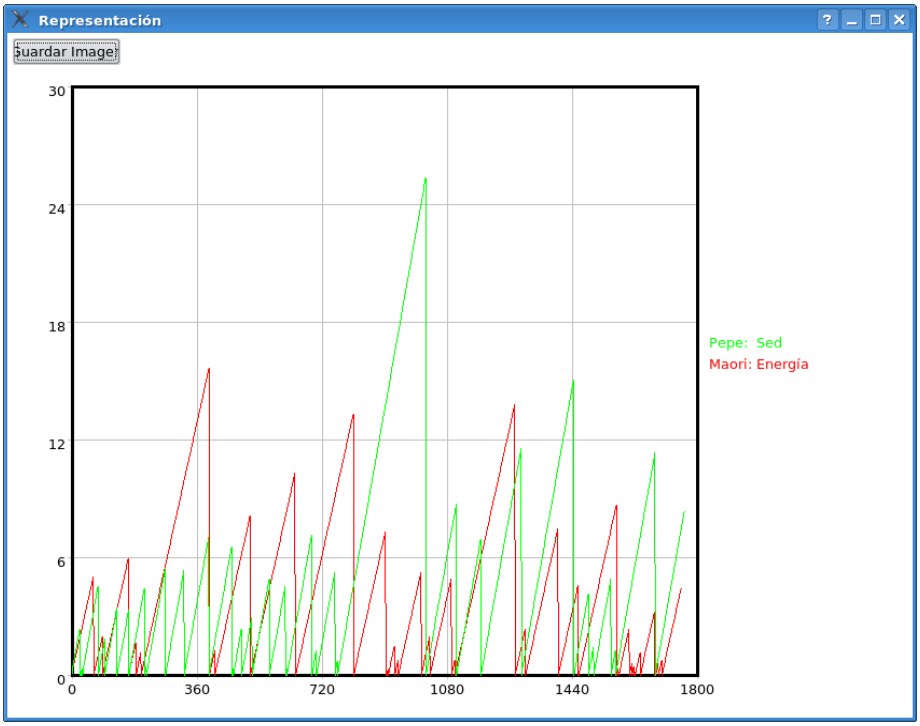

Figura 5.13 Gráfica de Energía (Maori), Sed (Pepe)

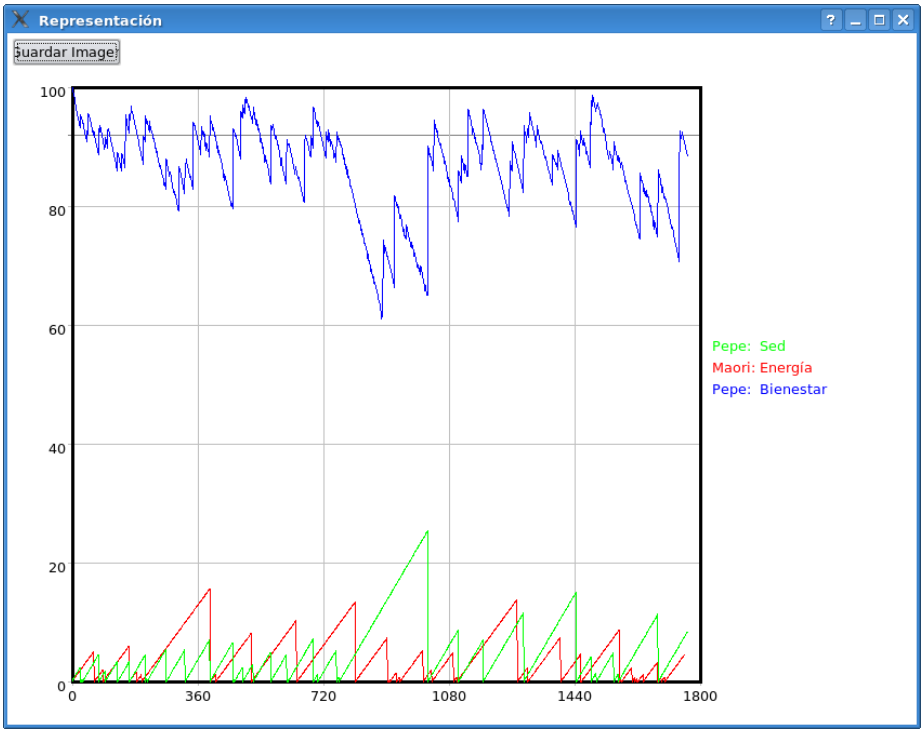

Figura 5.14 Gráfica de Energía (Maori), Sed (Pepe) y Bienestar (Pepe)

También se pueden añadir variables cuyos datos sean negativos. Se añadirá una recta de color gris correspondiente al valor del "cero" y la gráfica se reajustará para dar cabida a los nuevos valores. Esto puede verse en la figura 5.15, donde añadimos las variables Qcomer_sc del agente Pepe.

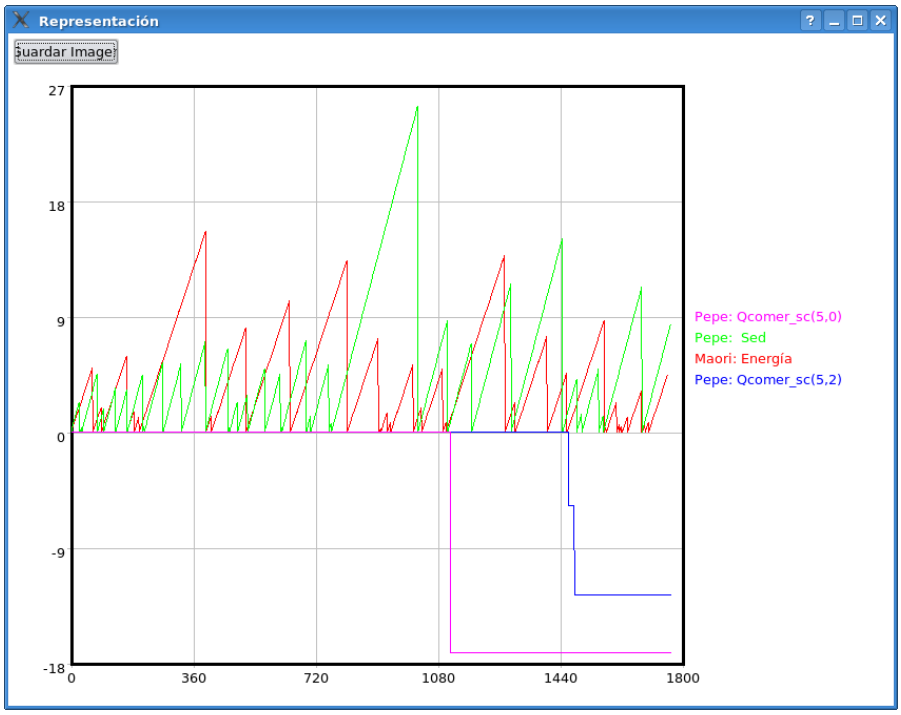

Figura 5.15 Gráfica de Sed (Pepe), Energía (Maori) y Q comer (Pepe)

Por último, pulsando el botón guardar imagen que se aprecia en todas las gráficas en la esquina superior izquierda, se desplegarán dos opciones BMP y JPEG, que corresponden a las extensiones de dos tipos de formatos de imágenes (ver Figura 5.16).
Los archivos con extensión BMP en los sistemas operativos windows, representan la sigla BitMaP (Bit Mapped Picture) mapa de bits. Es el formato propio del programa Microsoft Paint, que viene con el sistema operativo Windows. Puede guardar imágenes de 24 bits (16,7 millones de colores), 8 bits (256 colores) y menos.

JPEG son las siglas de "Joint Photographic Experts Group" nombre de la comisión que creó la norma. Es un método comúnmente utilizado para la compresión de imágenes fotográficas. El grado de reducción se puede ajustar, lo que permite seleccionar el compromiso que existe entre el tamaño de almacenamiento y la calidad de la imagen. Normalmente alcanza una compresión de 10 a 1 con pocas pérdidas perceptibles en la calidad de la imagen.

Una vez elegido el formato con el que se guardará la imagen aparecerá la figura 5.17. En ella hay que poner el nombre que tendrá el archivo de la imagen. Una vez escrito el nombre y seleccionada la ruta en la que se guardará el archivo procederemos a pulsar el botón "Save" y se guardará la imagen en el disco duro del ordenador.

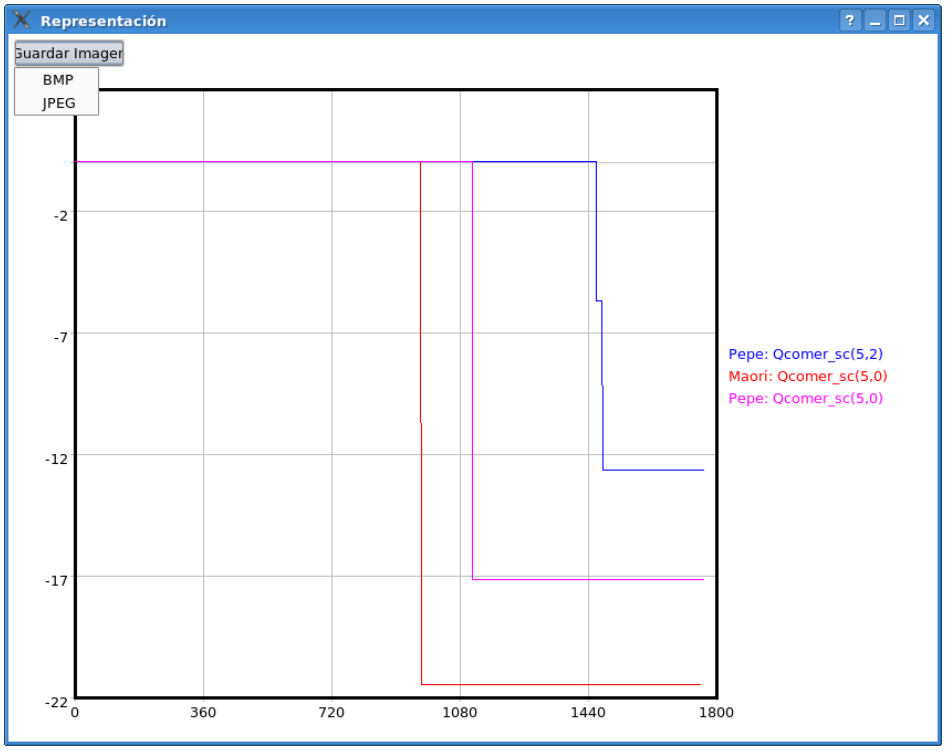

Figura 5.16 Gráfica con formatos BMP y JPEG

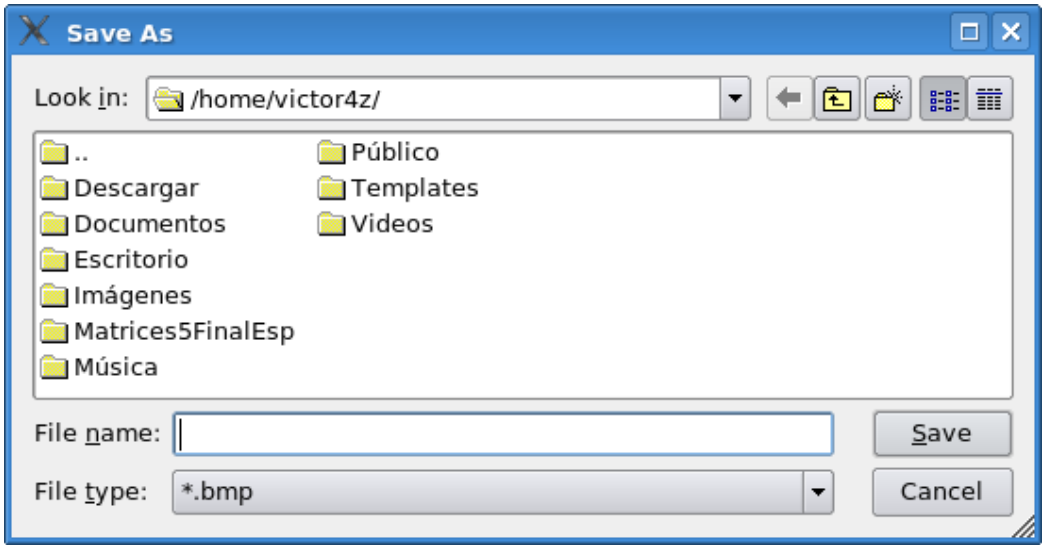

Figura 5.17 Pantalla para salvar archivo

Capítulo 6: CONCLUSIONES Y TRABAJOS FUTUROS

6 CONCLUSIONES Y TRABAJOS FUTUROS

6.1 Conclusiones del proyecto

El principal objetivo de este proyecto ha sido desarrollar una serie de herramientas que sirvan de ayuda en el análisis del comportamiento de un agente autónomo.

Para ello se han realizado los siguientes desarrollos:

- Creación de un Mundo Virtual con dos posibles entornos experimentales, (Passage y Castle), donde puede desarrollarse la vida de los agentes así como su proceso de aprendizaje.

- Desarrollo de un Sistema de Almacenamiento de Datos. Este sistema constituye la "memoria" de los agentes donde se guarda el resultado de las partidas que constituyen sus vivencias.

- Creación de una Interfaz Gráfica que permite la visualización de los datos y resultados para su posterior evaluación.

Los Mundos Virtuales creados en este proyecto, (Passage y Castle), constituyen un entorno idóneo para simular la "vida" de un robot o agente en su proceso de aprendizaje y pueden ser una base para desarrollos y estudios posteriores.

Dichos Mundos Virtuales se han creado basándose en la utilización de MUD (Dominios Multi Usuarios), con el objetivo de crear un entorno complejo de experimentación y aprendizaje del agente. La razón por la que se ha elegido la opción de un juego basado en texto ha sido la sencillez para enviar y recibir información del juego.

Entre diferentes códigos de MUD's disponibles en la red, se ha elegido para la implementación de este proyecto el denominado CoffeeeMud, basado en Java, por estar perfectamente documentado el software y disponer de manuales de uso claros.

En el proceso de vida de un agente autónomo se necesita un sistema de almacenamiento de datos que "memorice" los acontecimientos y situaciones que afectan al agente. En nuestro caso dichos datos son variados y complejos: datos de las partidas, datos de cada jugador, valores Q, etc.

En este proyecto el Sistema de Almacenamiento de Datos se ha basado en la utilización de una base de datos relacional. Para la aplicación a este proyecto se ha elegido MySQL como sistema de gestión de Base de Datos relacional, por su potencia, su facilidad de uso y la disponibilidad de documentación.

MySQL es un sistema de administración relacional de bases de datos para múltiples usuarios que utiliza el lenguaje SQL estandarizado para el almacenamiento, actualización y acceso a información.

La visualización de los datos es imprescindible para el análisis de los resultados del proceso de vida y aprendizaje de un agente. Para ello se ha desarrollado una Interfaz Gráfica que permite visualizar cualquier dato y su evolución a lo largo de la partida.

La Interfaz Gráfica se ha desarrollado mediante la herramienta Qt. Qt se puede definir como una herramienta software utilizada para la creación de Interfaces Gráficas de Usuario, (GUI). Consiste en un conjunto de librerías multiplataforma para el desarrollo de aplicaciones GUI escritas en C++, lo que indica que está orientada a objetos.

Las razones de su elección han sido el disponer de una librería para interfaces gráficos, estar orientada a objetos, facilidad de uso que permite el uso de ventanas, así como su versatilidad en la representación de los resultados correspondientes a los parámetros más relevantes en la vida del agente: Energía, salud, valores Q. Todo ello en beneficio de un mejor análisis de dichos parámetros y de su evolución en el tiempo.

6.2 Trabajos futuros

Como es un proyecto orientado a la creación de unas herramientas de ayuda a otros proyectos globales, sus posibles mejoras y trabajos futuros dependerán de las necesidades de dichos proyectos.

A título de ejemplo, se apuntan a continuación alguna de las posibilidades de mejora del proyecto:

- Los Mundos Virtuales, (Passage y Castle), podrían mejorarse creando mundos más complejos con más dependencias, más objetos y mejores imágenes, similares a las de los juegos en tres dimensiones, mediante la utilización de herramientas más potentes y sofisticadas.

- La Interfaz Gráfica permite posibilidades de ampliación casi ilimitadas en cuanto al número de variables. Se podrían introducir nuevos métodos estadísticos, (p.e. la mediana), así como gráficos en tres dimensiones que potenciaran la visualización de los resultados.

REFERENCIAS

7 REFERENCIAS

- Angulo Usategui José Mª , Susana Romero Yera, Ignacio Angulo Martínez (2005), *"Introducción a la Robótica",* Ed. Thomson.

- Arkin, R. C. (1988) "*Homeostatic control for a mobile robot: Dynamic replanning in hazardous environments.*" In SPIE Conference on Mobile Robots, Cambrige, MAA.

- Blanchette Jazmín , Mark Summerfield (2004), *"C ++ Programming with Qt 3"*.

- Barnetck, Christoph, Forlizzi, Jodi (2004), *"Shapin Human Robots Interaction: Understanding the Social Aspects of Intelligent Robotics Products"*.

- Bellman, K. L. (2003) "*Emotions in Humans and Artifacts, chapter Emotions: Meaningful mappings between the individual and its world*", MIT Press.

- Cañamero, L. (2003) *"Emotions in Humans and Artifacts, chapter Designing emotions for activity selection in autonomous agents"*, MIT Press.

- Charte Ojeda Francisco (2005), *"Guía Práctica de Usuarios SQL"*, Ed. Anaya.

- Gadanho, S. (1999), "*Reinforcement Learning in Autonomous Robots: An Empirical Investigation of the Role of Emotions",* Ph.D.thesis, University of Edinburgh.

- Isbell, C., Shelton, C. R., Kearns, M., Singh, S., and Stone, P. (2001). "*A social reinforcement learning agent. In the fifth international conference on Autonomous agents"*. Montreal, Quebec, Canadá.

- Malfaz María, Miguel A. Salichs (2004), *"A New Architecture for Autonomous Robots Based on Emotions"*.

- Malfaz María (2007), *"Sistema de toma de decisiones basado en emociones y autoaprendizaje para agentes sociales autónomos"*, Tésis doctoral.

- Norton Peter, *"Introducción a la Computación",* Ed. Mc Graw Hill, (2006).

- Thomaz, A. L. and Breazeal, C. (2006), *"Transparency and socially guided machine learning"*. In the 5th International Conference on Developmental Learning (ICDL)".

- Watkins, C. J. (1989). *"Models of Delayed Reinforcement Learning"*. Ph.D.thesis, Cambridge University, Cambridge, UK.

- Página de coffeemud: http://www.zimmers.net/home/mud/

- Página de las librerías mysql plus plus: http://tangentsoft.net/mysql++/

- Página de la wikipedia: http://es.wikipedia.org/

- Página del software qt: http://www.qtsoftware.com/
- Documentación de herramienta qt: http://doc.trolltech.com/