Capítulo 5

Conocimiento y Personalidad

La simulación realista del comportamiento de los agentes virtuales autónomos implica que ellos deben ser capaces de realizar una exploración inteligente del entorno que los rodea. Explorar *inteligentemente* el entorno se refiere a cumplir con los siguientes cuatro pasos:

1. Obtener información del ambiente, identificando los diferentes objetos del mundo virtual (*reconocimiento del ambiente*).
2. Transformar la información en conocimiento (*adquisición del conocimiento*).
3. Tomar decisiones basadas en dicho conocimiento y en los rasgos propios de "personalidad" del agente (*definición y planificación de objetivos*).
4. Llevar a efecto las acciones necesarias para el logro de los objetivos (*ejecución de acciones*).

Los puntos 1 y 4 fueron cubiertos en la Sección 4.3 del capítulo anterior, dedicado al Sistema de Control Físico de MaGeS.
Los puntos 2 y 3 constituyen el verdadero núcleo del proceso de control del comportamiento del AVA, ya que ellos están asociados a procesos cognitivos humanos (como el aprendizaje, memoria, razonamiento, etc). Entre mejores sean las técnicas empleadas para emular estos procesos, mayor será la credibilidad lograda en la toma de decisiones del agente; consiguiendo además, comportamientos cada vez menos predecibles, propios del ser humano, lejos de las reacciones mecánicas exhibidas por otros sistemas de agentes virtuales autónomos.

Hasta ahora, no son muchos los aportes significativos al respecto desde un punto de vista de la computación gráfica; sin embargo, podemos destacar los trabajos [FUNG99, GRAN95, GRZE98, THAL01, VAN93], donde las técnicas de Inteligencia Artificial (tales como redes neuronales y sistemas expertos) juegan un rol primordial, ya que ellas se basan en la idea de reproducir la estructura y comportamiento del cerebro humano.

En este capítulo se hace una introducción al Sistema de Control del Comportamiento de MaGeS (Figura 4.1), en el cual se realizan los pasos 2 y 3 previamente comentados. Se hace especial énfasis en los módulos correspondientes a la adquisición del conocimiento y a los atributos de personalidad del agente, dejando para el siguiente capítulo lo concerniente a la toma de decisiones. Con esto, se busca ofrecer una mejor descripción del sistema y de las técnicas que se implementan, ya que es aquí donde se concentran los principales aportes de esta investigación.

5.1 Sistema de Control del Comportamiento

Como se ha podido ver en el capítulo anterior, el marco de simulación propuesto (MaGeS) divide el diseño del agente virtual en dos sistemas bien definidos, el Sistema de Control Físico y el Sistema de Control del Comportamiento (Figura 4.1).

Si bien MaGeS, y en particular el Sistema Agente Virtual Autónomo, ofrecen los lineamientos a seguir para el desarrollo de actores virtuales autónomos en general (humanos, animales), habitando ambientes 3D; el diseño del Sistema de Control del Comportamiento (SCC) esta especialmente pensado para simular comportamientos complejos, como los que se esperan de un humano virtual.

La Figura 5.1 muestra el SCC (tomada de la Figura 4.1 de la arquitectura MaGeS), y en la que se aprecian sus componentes y flujo de información.

El Sistema de Control del Comportamiento representa el cerebro del AVA, y en él se reúnen varias técnicas que emulan en diversos grados algunos de los procesos cognitivos humanos. Con ello se pretende dotar al agente virtual con atributos mentales que lo capaciten para tomar decisiones en respuesta a los cambios del entorno; decisiones que deben ser consistentes con su condición de “imitador” de un ser humano.

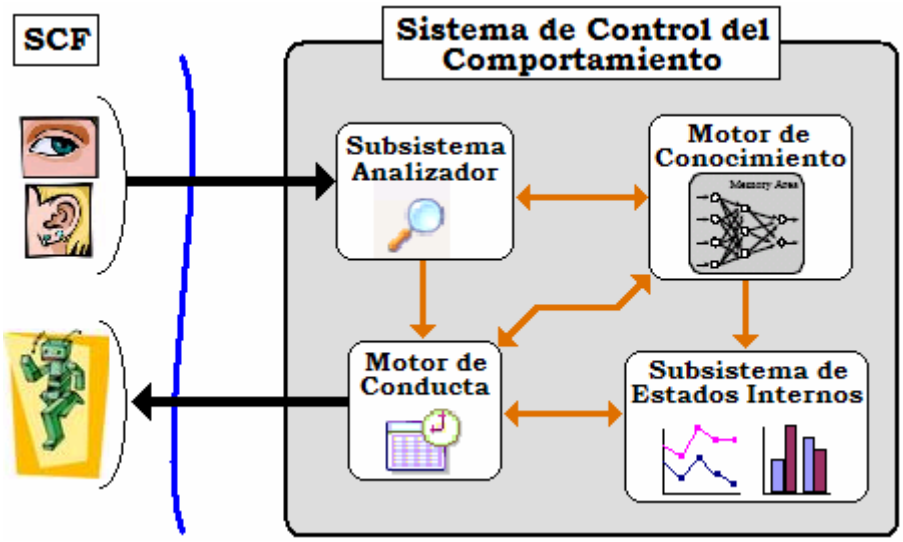

Figura 5.1. Sistema de Control del Comportamiento

El SCC es un sistema independiente, en el sentido de que puede implementarse en una plataforma distinta al resto de los componentes de MaGeS, siempre y cuando se respete un claro protocolo de comunicación entre los sistemas involucrados (ver Sección 4.1.2.1). Del mismo modo, su desarrollo no depende de la finalización de otros sistemas, como se verá mas adelante.

El Sistema de Control del Comportamiento esta formado por cuatro módulos (Figura 5.1), cuyas funciones abarcan las siguientes:

1. Transformar información en conocimiento. Esto se refiere a establecer relaciones entre los datos asociados a una cosa o hecho, de manera de poder distinguir dicha cosa o hecho del resto.
2. Administrar los atributos de personalidad. Esto se refiere al empleo de parámetros y variables que emulan los cambios anímicos que afectan la toma de decisiones.
3. Controlar la memoria. Esto se refiere a simular los procesos relacionados a la memoria humana, como lo son el recuerdo y el olvido.
4. Definir la factibilidad de los objetivos. Esto se refiere a establecer las probabilidades de éxito de los objetivos que el agente puede pretender alcanzar.
5. Mantener un objetivo a lograr. Se refiere a que en cada instante de tiempo el agente se encuentra realizando o intentando alcanzar un objetivo determinado, por lo que resulta indispensable la persistencia, factibilidad y consistencia en la designación del objetivo a perseguir.
6. Determinar un plan de acción. Esto se refiere a definir la estrategia de acciones a realizar para el logro efectivo del objetivo.

Todas esas tareas tienen una única finalidad: ofrecer las directrices para expresar físicamente una respuesta "razonada" a los cambios externos e internos que experimenta el agente. El SCC determina las directrices, y el SCF las transforma en acciones físicas; para ello, se establece un protocolo de comunicación entre ambos sistemas, que define las características del envío y recepción de los mensajes entre ellos.

5.1.1 Protocolo de Comunicación

Para tomar decisiones, el Sistema de Control del Comportamiento requiere información del entorno del AVA, la cual obtiene del Subsistema de Percepción del Sistema de Control Físico (Figura 5.1). Las especificaciones de este envío de información son las siguientes:

a) Si la percepción es producida por la visión virtual

- Número asociado a la visión.
- Cromosoma identificador del elemento visto (ver Sección 4.3.1.1).
- Coordenadas del objeto dentro del ambiente virtual.
- Dimensiones del objeto visto.
- Si es un objeto inteligente: el estatus (activado o no), disponibilidad (si se puede usar o no), etc.
- Código que indica si el objeto fue visto dentro del área de visión clara o difusa (Sección 4.3.1.2).

b) Si la percepción es producida por la audición virtual

- Número asociado a la audición.
- Código del mensaje (si es una frase, ruido o sonido, etc.
- Código del contenido (identificador de la frase o el ruido).
- Cromosoma del emisor del sonido.
- Coordenadas del emisor.

Una vez procesada internamente dicha información, el SCC envía al Subsistema de Movimiento del SCF ordenes concretas relacionadas a los requerimientos físicos (ver Sección 4.3.2) que el agente necesita ejecutar para el logro de sus objetivos.

La comunicación entre el Sistema de Control del Comportamiento y el Subsistema de Movimiento se realiza a través de dos variables del espacio de memoria del SCC, las cuales son `status` y `comando`. Estas variables son

actualizadas por el SCC en cada ciclo de animación del agente, y consultadas por el Subsistema de Movimiento para reflejar externamente su situación y postura.

El Sistema de Control del Comportamiento lleva un registro de la actividad que se encuentra realizando el agente, en la variable `status`; actividades como estar quieto, caminando, sentado, leyendo, ejercitándose, conversando, jugando, etc. Esta información no solo es importante para llevar control de las acciones físicas del agente, sino que además facilita identificar su actividad actual como requerimiento del Subsistema de Percepción de otros agentes que lo estén observando.

Con ayuda de la variable `comando` se pueden lograr conductas un poco mas elaboradas, como la de mirar a los lados mientras se esta caminando. Entre las ordenes que envía el SCC al Subsistema de Movimiento (Sección 4.3.2) a través de esta variable, se encuentran: mirar en una cierta dirección (ver Figura 4.9), mover los brazos de determinada manera, sentarse en un lugar especifico, pararse recto, activar o desactivar un objeto inteligente, abandonar un asiento, decir alguna frase, etc.

5.1.2 Sistema Independiente

Hasta ahora hemos visto como se relacionan el Sistema de Control del Comportamiento y el Sistema de Control Físico, y como se establece la comunicación entre ellos; sin embargo, la estructura y funciones de dichos sistemas permiten su desarrollo de manera independiente, es decir, que no es necesario completar un sistema para iniciar el desarrollo del otro, y pueden ser ejecutados y probados separadamente.

Por ejemplo, para desarrollar y realizar evaluaciones preliminares de los diferentes módulos que integran el SCC, de manera independiente, los datos de entrada que necesita el sistema pueden ser proporcionados de manera controlada para ir midiendo el desempeño de los algoritmos programados. En este sentido, el lenguaje PROLOG y el entorno de programación Amzi!Prolog (herramientas con las que fue desarrollado el SCC, ver Sección 4.1.1) constituyeron una excelente elección al momento de las pruebas preliminares por ser Prolog un lenguaje interpretado por naturaleza (aunque la versión de Amzi puede ser compilada), ya que permiten la manipulación dinámica y en tiempo de ejecución de los datos de memoria. La Figura 5.2a muestra el ingreso manual de percepciones para las pruebas del SCC; aunque estas también pueden prepararse en un archivo de texto para un ingreso mas rápido. La Figura 5.2b muestra los resultados de la salida del

sistema que permite monitorear su ejecución. Estas figuras ilustran como se desarrolla y prueba el SCC, ajeno al entorno Visual C++ (plataforma bajo la que se desarrollo el resto de software de animación).

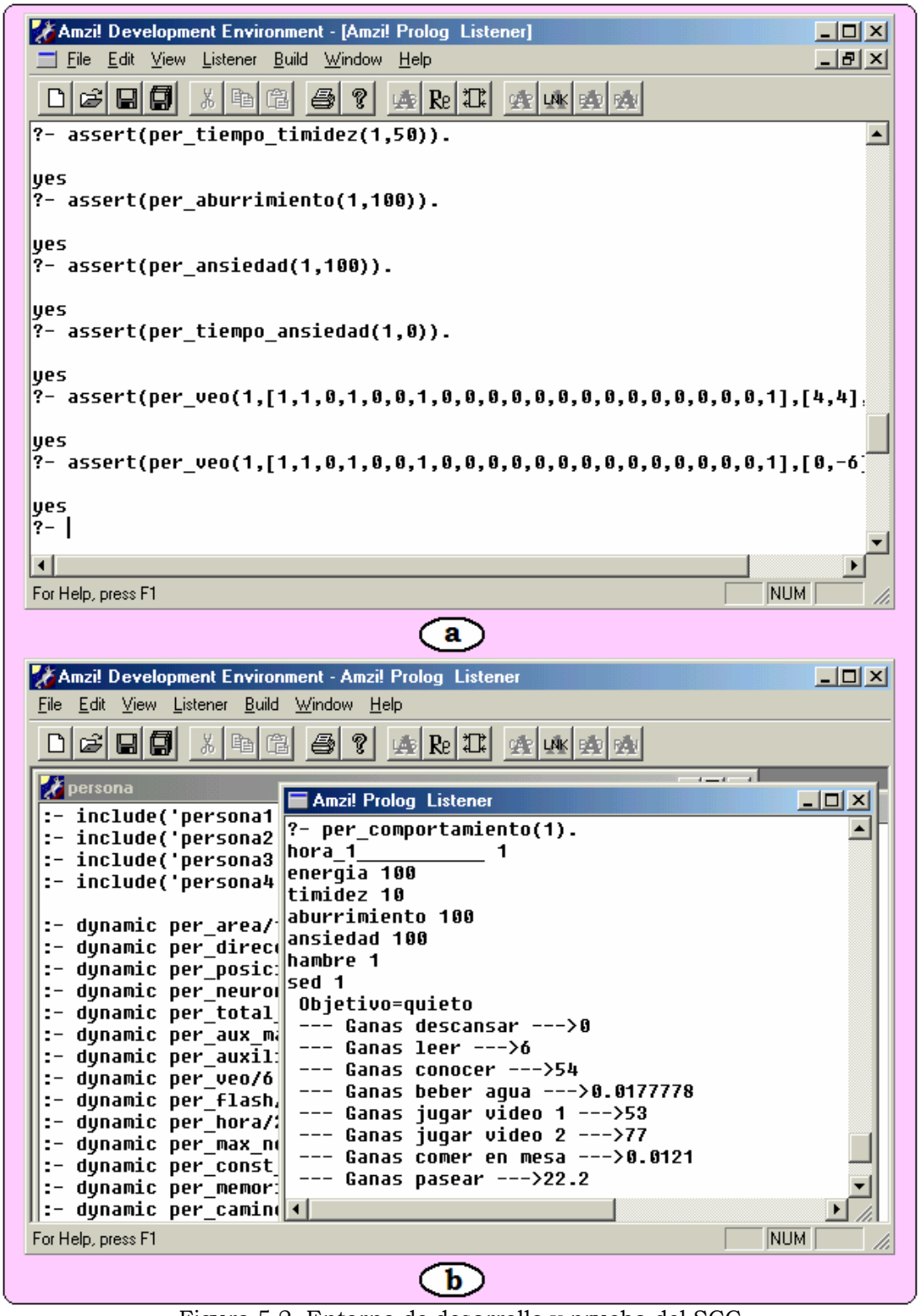

Figura 5.2. Entorno de desarrollo y prueba del SCC

La independencia de ejecución del SCC con respecto al resto de los módulos que conforman la arquitectura MaGeS facilita sus pruebas; sin embargo, para una definitiva validación de las rutinas implementadas, lo mejor es conectar todos los componentes de MaGeS y ejecutarlos bajo escenarios específicos de prueba.

En resumen, el diseño independiente del Sistema de Control del Comportamiento (Figura 5.1) ofrece las siguientes ventajas:

- Reducción de los tiempos de desarrollo, ya que puede trabajarse en paralelo con el resto de los sistemas que conforman MaGeS.
- Fácil actualización, ya que su lógica de programación no afecta otros sistemas, por lo que resulta mas cómodo encontrar los errores e incorporar nuevas rutinas sin tener que modificar el resto del programa.
- Versatilidad, ya que puede ser incorporado en otros entornos sin ningún o casi ningún cambio; es decir, que se puede cambiar el diseño físico del agente y del ambiente 3D, conservando el mismo SCC.

Los módulos que conforman el SCC son descritos en las siguientes secciones, salvo el Motor de Conducta, al cual se le dedica un capítulo aparte.

5.2 Motor de Conocimiento

El primer módulo que se describe es el Motor de Conocimiento, ya que representa el centro de información de todo el SCC. Si partimos de que todo ser humano es el resultado de sus vivencias y recuerdos, puede intuirse la importancia de este módulo, y por qué el resto de las componentes del SCC requieren de comunicación con él (Figura 5.1).

El Motor de Conocimiento reúne varias técnicas de Inteligencia Artificial para emular las capacidades cognitivas asociadas con la *memoria humana*. Su finalidad dentro del Sistema de Control del Comportamiento es almacenar y actualizar el conocimiento y los recuerdos del agente; es decir, toda información exterior que recibe el agente es guardada, y al hacerlo pasa a formar parte de sus recuerdos, y la exactitud de estos dependerá (entre otros factores) de la antigüedad de los mismos. Adicionalmente, el Motor de Conocimiento asocia información relacionada entre sí para generar nuevo conocimiento, lo que le proporciona al agente una mejor “comprensión” del entorno y sus elementos.

La Figura 5.3 muestra la estructura del Motor de Conocimiento, la cual incluye tres componentes de almacenamiento (la Zona Flash, el Área de Memoria, y la Base de Conocimiento) y dos módulos de procesamiento (el Actualizador de Conocimiento y el Resolvedor de Peticiones).

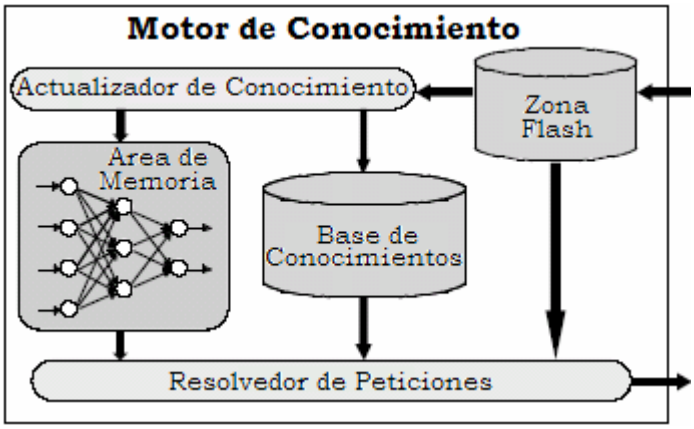

Figura 5.3. Estructura del Motor de Conocimiento

5.2.1 Zona Flash

La Zona Flash es un buffer de memoria donde se almacena toda la información suministrada por el Sistema de Control Físico (las percepciones).

En cada instante de tiempo el SCF recoge información visual y auditiva del entorno, la cual transmite al SCC; estas nuevas percepciones son almacenadas temporalmente en la zona flash, hasta que nueva información sea percibida, reemplazando completamente todo lo que ella contuviese anteriormente.

Como puede notarse, este buffer de memoria es utilizado para llevar control de lo que el agente está percibiendo, y no de lo que ha percibido.

Por cada percepción, los datos almacenados en la Zona Flash son los siguientes:

- Cromosoma identificador del elemento (ver Sección 4.3.1.1).
- Información relacionada al elemento, como posición, estatus, etc (ver Sección 5.1.1).
- Hora (tiempo) en que se origina la percepción (ver Sección 5.3 para detalles).
- Nivel de impacto asociado a la percepción (ver Sección 5.3 para detalles)

5.2.2 Área de Memoria

El Área de Memoria es la zona reservada para almacenar los recuerdos del agente virtual.

Como se comentara anteriormente, los recuerdos se construyen y arraigan a partir de las percepciones del agente, y estos le ayudan a conocer su entorno e interactuar con él. Una manera simple de hacerse de dichos recuerdos es almacenando en el Área de Memoria toda nueva percepción que reciba. Pero esto no es suficiente, ya que este enfoque presenta la limitante de que no considera la capacidad de olvido, permitiendo que el agente pudiese ubicar fácilmente elementos dentro del entorno aunque estos los hubiese visto muy brevemente mucho tiempo atrás; lo cual no seria una situación muy creíble, ya que todos los recuerdos mantienen el mismo nivel de detalle y no desaparecen durante la vida del agente.

La manera de solventar la carencia de "olvido" del enfoque anterior es mediante un esquema de *memoria de corto, mediano y largo término* (memoria CML). El enfoque CML consiste en incorporar a la información asociada con cada recuerdo, dos parámetros adicionales; el primero representando el nivel de impacto de la percepción (el cual marcará la duración del recuerdo en memoria, para ser olvidado a un corto, mediano o largo plazo), y el segundo, el tiempo trascurrido desde la ultima vez que el recuerdo fue actualizado; es decir, reconsiderar el tiempo asociado al recuerdo cada vez que una percepción refuerce su permanencia en memoria. Un enfoque muy similar fue utilizado en [CAIC00].

Con dicha información es posible saber la antigüedad de cada recuerdo, y en qué momento estos deben borrarse de memoria. Además, el enfoque CML implementa el concepto de reforzamiento, al actualizar como reciente un recuerdo (que lleve cierto tiempo en memoria) cuando se recibe nueva información relacionada a él. Sin embargo, este esquema de memoria añade dos consideraciones importantes para su uso. Se deben establecer los criterios para determinar el nivel de impacto de cada percepción (esto lo realiza el Subsistema Analizador, Sección 5.3), y los criterios para el uso efectivo de los recuerdos según su antigüedad en memoria, ya que en la realidad humana sucede que entre mas cerca este un recuerdo de ser olvidado, mas imprecisos podrían ser los datos asociados a él; y esto no sucede con la información almacenada en un ordenador. Una manera de resolver esto es añadir "ruido" a los datos del recuerdo a medida que pase el tiempo y estén más cerca del olvido.

Al emular la capacidad de memoria humana, en un agente virtual, se pretende que este exhiba las siguientes características en su comportamiento (aunque se hará referencia a lugares, también se aplica a objetos y otros agentes):

- Que al recordar los lugares visitados, pueda regresar a ellos.
- Que demuestre inseguridad al intentar regresar a un lugar que casi no recuerda.
- Que olvide lugares brevemente vistos, y recuerdos con mucho tiempo sin rememorar.
- Que algunos agentes muestren tener más memoria que otros.

Si bien estas características pueden lograrse con el esquema de memoria CML, se ha ideado un novedoso enfoque que lo realiza de manera natural, y que se discute a continuación.

5.2.2.1 Memoria Neuronal

El Área de Memoria de la Figura 5.3 implementa lo que denominaremos una *Memoria Neuronal*; es decir, una red neuronal [HAYK94, HERT91] cuyas neuronas intentan almacenar todos los elementos percibidos en cada instante de tiempo.

Todo agente virtual autónomo es implementado con un número predefinido de neuronas, las cuales determinan la capacidad de memorización del agente. La Figura 5.4a ilustra la estructura de una neurona (lo que corresponde a su vector de pesos).

Además de la información a ser recordada (segundo y tercer campo de su vector de pesos), las neuronas tienen asociado un identificador (para tener acceso a cada una de ellas de manera inequívoca), un valor para el último instante de tiempo en que fueron actualizadas (con lo que se determina su actividad), y un numero que define su velocidad de aprendizaje (es decir, del tiempo que requeriría para memorizar una percepción).

La Red Neuronal que forma el Área de Memoria es una red auto asociativa [KOHO97], cuyas neuronas intentan adaptarse a todas las entradas recibidas, tal como se muestra en la Figura 5.4c, donde se ilustra el progreso de 2 neuronas intentando reducir el error entre las entradas y sus pesos. La Figura 5.4b muestra la estructura de la red. La Capa de Entrada representa solo el paso de los patrones de entrenamiento; y la Capa de Competición representa la red en sí, y es donde se encuentran las

neuronas. En nuestra implementación, no se establecieron relaciones de vecindad entre las neuronas, por lo que la adaptación de una no afecta a las otras (para mas detalles sobre esto, ver en la Sección 5.2.4.1, dedicada al método de entrenamiento de la red).

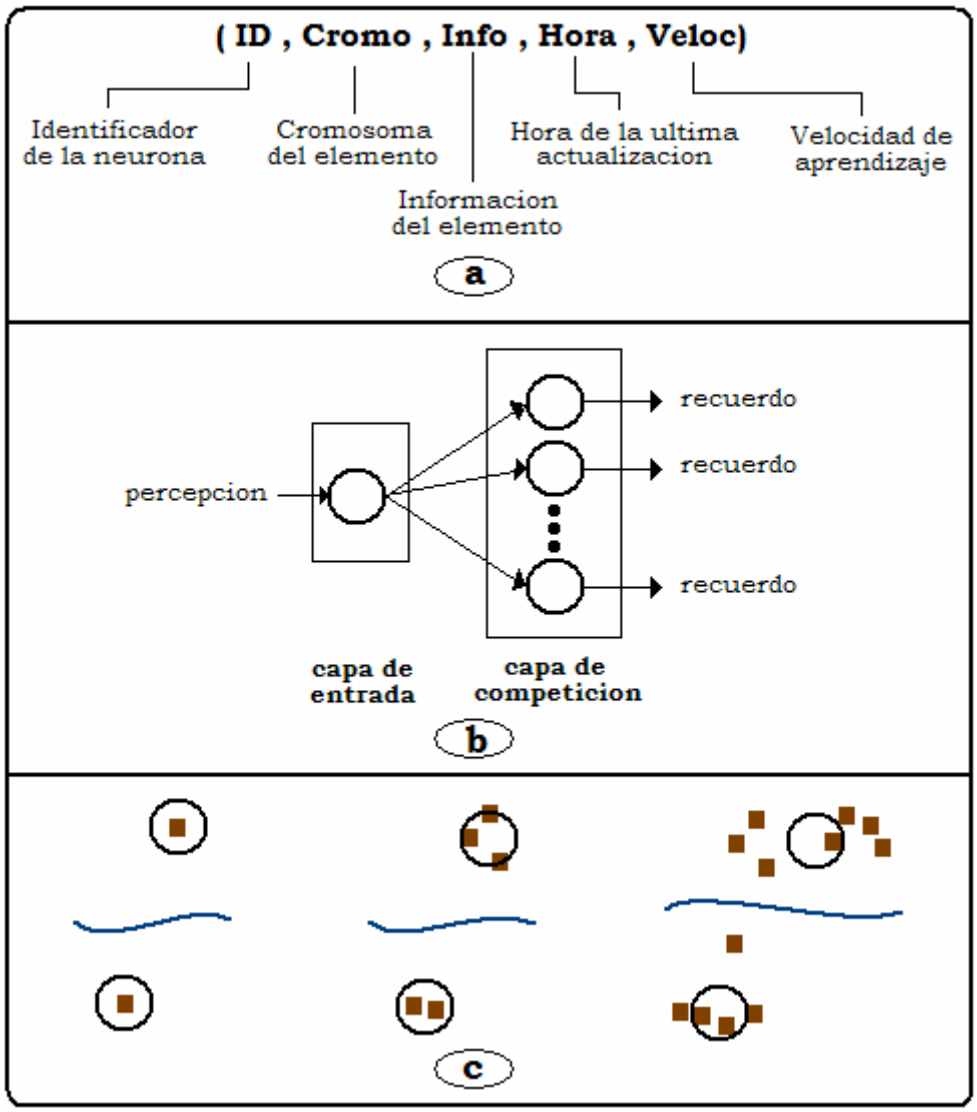

Figura 5.4. a) Estructura de una neurona. b) Separación en clusters

Para el caso que nos ocupa, se pretende que un pequeño número de neuronas almacenen las percepciones que el agente virtual va teniendo a lo largo de su ciclo de animación.

Cada cromosoma de un elemento percibido ocupa una posición en un espacio de cromosomas. Lo que la red intenta hacer es subdividir dicho espacio en grupos (tantos como neuronas hayan), con el propósito de asociar cada cromosoma a una neurona especifica (la mas cercana a ella); tal como se muestra en la Figura 5.4c. Con esto, la red pretende minimizar

la suma de los cuadrados de los residuales internos de cada grupo; los cuales son básicamente las distancias de la localización de cada cromosoma al respectivo centroide de su grupo.

A la final, la información contenida dentro de cada neurona es el resultado de la ponderación de todos sus datos de entrada. El campo asociado a la velocidad de aprendizaje (Figura 5.4a) es determinante en este sentido, ya que actúa como un peso que acelera o desacelera el tiempo de adaptación de la neurona a la nueva información percibida; por ejemplo, si su velocidad de aprendizaje es 0, actúa como un inhibidor de aprendizaje, imposibilitando a la neurona a adaptarse a nuevas entradas, lo que se interpreta como que dicha neurona almacena un recuerdo "inolvidable". El valor por defecto para este parámetro es el 1, el cual equivale a una velocidad normal de adaptacion. Más detalles sobre este campo en la Sección 5.2.4.1.

El siguiente ejemplo ilustra el poder de representación del esquema propuesto.

Utilizando el esquema de codificado de la Tabla 4.1 (de la Sección 4.3.1.1), supongamos que tenemos una neurona que almacena el cromosoma de un ave pequeña en el instante 0, sin preocuparnos del resto de la información asociada a dicho cromosoma; tal como se muestra a continuación,

(1 , [0,1,0, 1,0,0, 0, 0,0,0,0,1] , _ , 0 , 1)

Ahora, supongamos que el AVA comienza a percibir a un perro grande durante un lapso *n* de tiempo, dejando de ver el ave. Si la neurona 1 es elegida para recoger esta nueva información, la neurona busca adaptarse a los nuevos datos de entrada durante el periodo en que se produce la misma. Pasado dicho lapso de tiempo, el vector de pesos de la neurona 1 podría verse como sigue,

(1 , [0,1,0, 0.75,0.25,0, 0.25, 0,0,0,0,1] , _ , 0 , 1)

El claro recuerdo del ave, almacenado inicialmente en la neurona 1, va cambiando para converger a los datos que identifican al perro. Sin embargo, en este punto, la neurona 1 mantiene dos recuerdos con diferentes grados de exactitud. Utilizando la formula de error, introducida en la Sección 4.3.1.1, entre el cromosoma almacenado en la neurona y los cromosomas del ave y del perro, se puede interpretar como que la neurona contiene información de ambos, con un 83% y un 50% de confiabilidad, respectivamente.

Este ejemplo ilustra algunas características interesantes del esquema propuesto:

- Con pocas neuronas se puede almacenar una gran cantidad de información (recuerdos).
- Por su estructura, la Memoria Neuronal puede aumentar dinámicamente su tamaño (incrementando su número de neuronas) para una mayor capacidad de memorización, en caso de ser necesario.
- Si la neurona del ejemplo es sometida a la misma percepción por suficiente tiempo, fijaría con exactitud los datos del perro y perdería los del ave; por lo que el "olvido" es manejado de manera natural por la red neuronal.
- La interpretación de los datos requiere manejo de incertidumbre, lo que contribuye a comportamientos más realistas, ya que se debe tomar decisiones basadas en información imprecisa, tal como ocurre en la realidad.

5.2.3 Base de Conocimientos

Si nos detenemos a observar el tipo de información que se almacena en los distintos módulos del Motor de Conocimiento, destinados para tal fin, vemos que en la Zona Flash se retienen brevemente las percepciones recibidas, que sirven como entrada a las neuronas del Área de Memoria para ser memorizadas. Pero esta información no es suficiente, si se pretende conseguir conductas sociales complejas.

Tal como su nombre lo indica, la Base de Conocimientos es un Sistema Experto Basado en Reglas (ver [GIAR98]), el cual contiene, tanto, conocimiento concreto (hechos o evidencias), como conocimiento abstracto (reglas de inferencia).

Entre los hechos que se registran en la Base de Conocimientos se encuentran aquellos intrínsecos al propio agente, como lo son los relativos a su identidad (cromosoma, nombre, relaciones de parentesco, gustos, si es adulto o niño, sexo), a su condición física (resistencia a la fatiga, rango visual, número de neuronas, coeficiente de aprendizaje), y a su personalidad (parámetros de sociabilidad, dinamismo, constancia). Lo usual es que estos hechos sean estáticos, y se mantengan durante todo el ciclo de vida del agente.

El resto de los hechos son dinámicos; es decir, se originan y varían a medida que el agente explora su entorno e interactúa con él. Mediante un conjunto de reglas de inferencia, las percepciones y los hechos que posee el agente son utilizados para generar nuevo conocimiento.

Veamos un ejemplo, en el que se emplea la sintaxis de Prolog para mayor legibilidad [BRAT00]. Supongamos que tenemos 3 AVAs con el siguiente conocimiento sobre sus gustos:

AVA: Juan	AVA: Pedro	AVA: Maria
`mis_gustos(jugar).`	`mis_gustos(jugar).`	`mis_gustos(charlar).`
`mis_gustos(charlar).`		

Luego de que los 3 AVAs se conocieran y transmitieran sus gustos, sus respectivos conocimientos serian los siguientes:

AVA: Juan	AVA: Pedro	AVA: Maria
`mis_gustos(jugar).`	`mis_gustos(jugar).`	`mis_gustos(charlar).`
`mis_gustos(charlar).`	`gustos(Juan,jugar).`	`gustos(Juan,jugar).`
`gustos(Pedro,jugar).`	`gustos(Juan,charlar).`	`gustos(Juan,charlar)`
`gustos(Maria,charlar).`	`gustos(Maria,charlar).`	`gustos(Pedro,jugar).`

Este nuevo conocimiento no es más que la transformación de percepciones (comunicación oral) en hechos. Sin embargo, el agente cuenta con conocimiento abstracto que puede utilizar para deducir conocimiento adicional al que posee actualmente. Mediante la siguiente regla de inferencia:

```
Afin(X):- mis_gusto(G), gustos(X,G), assert(amigo(X)).
```

cada agente establece un criterio de con quien goza de empatía; es decir, con quien comparte gustos y le es mejor estar. El resultado de esto es el siguiente:

AVA: Juan	AVA: Pedro	AVA: Maria
`mis_gustos(jugar).`	`mis_gustos(jugar).`	`mis_gustos(charlar).`
`mis_gustos(charlar).`	`gustos(Juan,jugar).`	`gustos(Juan,jugar).`
`gustos(Pedro,jugar).`	`gustos(Juan,charlar).`	`gustos(Juan,charlar)`
`gustos(Maria,charlar).`	`gustos(Maria,charlar).`	`gustos(Pedro,jugar).`
`amigo(Pedro).`	`amigo(Juan).`	`amigo(Juan).`
`amigo(Maria).`		

Este sencillo ejemplo es una muestra del tipo de información contenida en la Base de Conocimientos, y de cómo ésta es tratada. Por supuesto, es el número de reglas y el diseño del motor de inferencia lo que determinan la complejidad del sistema. Sin embargo, al emplear Prolog como lenguaje de desarrollo para este módulo, la única preocupación es definir y estructurar adecuadamente la base de conocimiento en sí (conjunto de hechos y reglas), ya que el motor de inferencia empleado es el mismo que el de Prolog.

Como se comentara al principio, hechos y relaciones como los mostrados en el ejemplo anterior ofrecen la oportunidad de desarrollar conductas sociales más elaboradas y creíbles. Entre mayor sea la diversidad de conocimiento empleado, mas amplia será la gama de posibles conductas explorables.

5.2.4 Actualizador de Conocimiento

Es el responsable de la actualización del Área de Memoria y la Base de Conocimientos.

Luego de que las percepciones son registradas en la Zona Flash, el Actualizador de Conocimiento activa los procedimientos de memorización del Área de Memoria y de generación de conocimiento de la Base de Conocimientos.

Para el caso de la actualización de la Base de Conocimientos, el Actualizador determina cuales percepciones debe transformar en hechos, y luego, hace un recorrido por todas las reglas asociadas al conocimiento abstracto para inferir nuevo conocimiento.

En el caso del Área de Memoria, el Actualizador implementa un algoritmo de aprendizaje no supervisado para el ajuste de las neuronas de la Memoria Neuronal. La siguiente sección explica dicho algoritmo.

5.2.4.1 K-Means

Toda red neuronal emplea un algoritmo de entrenamiento para ajustar los pesos de sus neuronas, de manera de minimizar una función de error asociada a la red. Según la finalidad de la red y de la información que se disponga, el algoritmo de entrenamiento puede ser o no, supervisado [FAUS94].

Dado que para la Memoria Neuronal (red neuronal del Área de Memoria, Sección 5.2.2.1) no se cuenta con un conjunto de patrones preestablecidos con los que la red aprenda (ya que seria como intentar conocer los recuerdos antes de tenerlos); sino por el contrario, los patrones de entrada van surgiendo según las vivencias que experimenta el agente y la red debe aprender de ellos, la técnica mas apropiada para la actualización de las neuronas de la Memoria Neuronal es mediante el algoritmo de K-Means, un esquema de aprendizaje no supervisado.

K-Means [BISH95, LEGE98] es un método de particionamiento mediante mínimos cuadrados, el cual permite dividir una colección de objetos en *k* grupos. El algoritmo itera entre dos simples pasos: 1) calcular el centroide de un grupo y usarlo como nueva semilla de grupo, y 2) asignar cada objeto al grupo de la semilla más cercana. Durante las iteraciones, el programa trata de minimizar la suma, sobre todos los grupos, de las distancias de los objetos a los respectivos centroides de su grupo. El particionamiento K-Means es conocido como un problema NP-completo, ya que no se puede garantizar alcanzar el mínimo global de la función objetivo.

El algoritmo k-Means se adapta perfectamente a los objetivos del Área de Memoria, ya que los patrones de entrada (las percepciones) suelen ser repetitivos, logrando el efecto de reforzamiento de los recuerdos. Por otro lado, aunque el espectro de percepciones es finito, el proceso de alimentación de la red no lo es, ya que solo cesa cuando la animación finaliza, con lo que la red se adapta a las percepciones mas recientes y frecuentes.

Aplicado a la Memoria Neuronal, el algoritmo K-Means es definido de la siguiente manera:

Notación: Sea P_k la primera de una serie de percepciones continuas del objeto *k*. PP_k la continuación de la percepción del objeto *k*, que se inició con P_k. Sirva M para denotar el conjunto de todas las neuronas, N_i para denotar a la neurona *i* sin ningún vinculo, y N_{ik} para designar a la neurona *i* vinculada al objeto *k*. Y finalmente, definimos $alfa_{ik}$ como la velocidad de aprendizaje (adaptación) de la neurona *i* al patrón de percepción *k*.

Algoritmo:
Paso 0: Se inicializan todas las neuronas, $N_i=0$.
Para cada percepción recibida de un objeto *k*
Paso 1: Para P_k y N_i entonces se vincula la neurona al objeto y $N_{ik}=P_k$.
Paso 2: Para PP_k y N_{ik} entonces se $N_{ik} = N_{ik} + alfa_{ik}*(PP_k-N_{ik})$

Paso 3: Si se recibe(P_k *o* PP_k) y no hay neuronas sin vinculo, entonces se elige N_{im} tal que sea $\min\{dist(N_{ij}, P_k)\}$, $\forall N_{ij} \in \mathrm{M}$, y $hora_i = \min\{hora_j\}$, $\forall N_j \in \mathrm{M}$. Se vincula la neurona N_{im} al objeto k, y $N_{ik} = N_{ik} + alfa_{ik}*(PP_k - N_{ik})P_k$

Paso 4: Se ajusta el valor de $veloc_i$ en caso de ser necesario.

Observaciones:

- El parámetro *alfa* estable la velocidad promedio de adaptación de las neuronas y depende del coeficiente de aprendizaje del agente (Sección 5.2.3), del parámetro *Veloc* propio de cada neurona (Figura 5.4a), y del nivel de impacto de la percepción (Sección 5.2.1). De allí el uso de los subíndices.
- En un comienzo las neuronas no almacenan ningún recuerdo (Paso 0).
- Al inicio, cada nuevo objeto de una percepción es asignado a una neurona (Paso 1).
- El Paso 2 pretende que cada neurona siga adaptándose al objeto asignado a ella, mientras lo siga percibiendo. Esto evita que si dos neuronas se encuentran adaptándose a percepciones parecidas, dichas percepciones se alternen entre las neuronas provocando un indeseado patrón de aprendizaje.
- $hora_i$ es el campo "hora" de la estructura de la neurona (Figura 5.4a).
- Como por lo general hay menos neuronas que objetos a memorizar, el Paso 3 define los criterios para elegir a la neurona que aprenderá del objeto que carece de vínculo. Para cada nuevo objeto percibido, se selecciona una neurona de entre aquellas que tengan mas tiempo sin ser actualizadas, y cuya distancia al objeto sea menor.
- El criterio de actualización del campo $veloc_i$ depende de un conjunto de simples reglas que son empleadas para determinar si la neurona almacenará un recuerdo inolvidable, o memorizara mas deprisa olvidando rápidamente lo anterior; o por el contrario podrá retener mas tiempo un recuerdo adaptándose mas lentamente a las nuevas percepciones.

Para mayor claridad en el funcionamiento del algoritmo, veamos un ejemplo.

La Figura 5.5 muestra la evolución de las 2 neuronas de un agente durante su desenvolvimiento en un parque con solo 3 objetos. Se utiliza un codificado simple (de un solo tramo) para facilitar la explicación (un video con esta simulación es incluido en el CD que acompaña este trabajo, en la sección Video - Ejemplos, número 10).

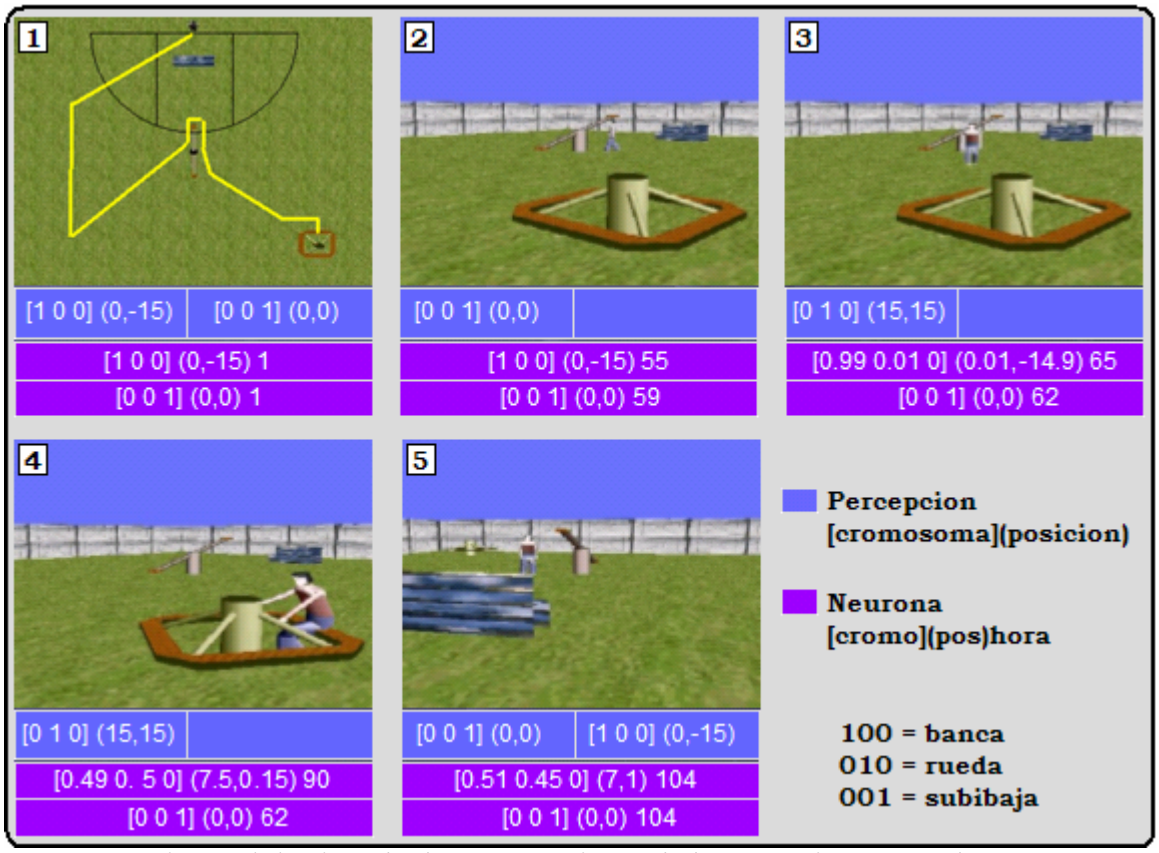

Figura 5.5. Ejemplo de entrenamiento de la memoria neuronal

La animación comienza con el objetivo inicial para el agente de jugar en la rueda. El cuadro 1 muestra todo el recorrido que hace el agente hasta encontrar la rueda, ya que inicialmente no la ve y tiene que buscarla.

Al iniciar la animación, el agente solo es capaz de percibir 2 objetos (la banca y el subibaja); con lo que sus neuronas son inicializadas a cada uno de ellos, tal como se muestra en el cuadro 1.

Hay un punto en el que el agente comienza a bordear el subibaja y deja de ver la banca (cuadro 2), teniendo solo la percepción del subibaja. Aunque ha dejado de ver la banca, tiene claro en su memoria el recuerdo de ésta, como lo muestra el contenido de sus neuronas.

Al seguir avanzando por un lado del subibaja, llega a ver la rueda, volteándose hacia ella y dándole la espalda a la banca y al subibaja (cuadro 3). Ahora la única percepción que recibe es la de la rueda, y k-means elige la neurona 1, que contenía el recuerdo de la banca, para ir almacenando este nuevo elemento. No se selecciona la otra neurona porque ella había sido actualizada más recientemente, como sus respectivas horas indican en el cuadro 2.

Mientras el agente solo ve la rueda e interactúa con ella, solo se actualiza la neurona 1 tal como se muestra en el cuadro 4.

Luego de mucho jugar el agente se cansa y su objetivo cambia a “descansar en la banca”. Como puede verse en sus neuronas (cuadro 4), el

agente recuerda perfectamente el subibaja, pero no así la banca de la cual solo tiene un 33.6% del recuerdo. La posición que almacena la neurona 1 es algún punto entre la rueda y la banca, por lo que el agente se dirige hacia ese punto (sabiendo que cerca de allí estará la banca). Ya mas cerca, llega un momento en que el agente alcanza a ver la banca y se dirige directamente a ella (cuadro 5).

El ejemplo muestra como las neuronas almacenan con distintos grados de certeza, información sobre los objetos percibidos; y como estos recuerdos pueden emplearse para el logro de los objetivos haciendo que el agente virtual exhiba comportamientos mas complejos (como el de la duda al buscar un lugar).

5.2.5 Resolvedor de Peticiones

Es el encargado de atender las solicitudes hechas al Motor de Conocimiento.

Como se ve en la Figura 5.1, todos los módulos del Sistema de Control del Comportamiento toman información del Motor de Conocimiento para sus procesos internos. Dicha información es solicitada al Motor de Conocimiento a través del Resolvedor de Peticiones, que es quien procesa el pedido y retorna la información deseada en el formato adecuado.

Seguramente, si el Motor de Conocimiento se hubiese implementado como una Base de Datos, utilizando un Sistema Manejador de Base de Datos, las peticiones las resolvería el propio manejador, y estas se harían mediante sentencias SQL, con toda la versatilidad que este lenguaje de consultas permite. Sin embargo, el haber desarrollado el Motor de Conocimiento en Prolog no solo permitió incorporar conocimiento abstracto, sino que además, también se cuenta con la versatilidad del lenguaje y de su motor de inferencia, para que al igual que con SQL podamos construir casi cualquier tipo de consulta.

El diseño del Resolvedor de Peticiones incluye una librería de consultas predeterminadas al que cualquiera otro módulo puede acceder, en función de la información y el formato de presentación que desee.

Es importante resaltar que los recuerdos almacenados en el Motor de Conocimiento pueden no ser totalmente veraces; es decir, que poseen grados de certeza. El Resolvedor incluye en su respuesta el porcentaje de exactitud

de la información que ofrece, para que quienes hayan hecho la petición tomen en cuenta la incertidumbre asociada.

5.3 Subsistema Analizador

Como se comentara anteriormente, el Sistema de Control Físico envía la información sobre las percepciones al Sistema de Control del Comportamiento, y es el Subsistema Analizador el responsable de recibir y procesar dicha información (Figura 5.1).

El Subsistema Analizador es un sistema experto con dos funciones básicas: 1) actualizar la Zona Flash, y 2) actualizar la factibilidad de los objetivos del agente.

Actualizar la Zona Flash

El Subsistema Analizador recibe dos tipos de percepciones, las directas y las indirectas.

Las percepciones directas son aquellas que llegan de manera nítida y clara al agente; por ejemplo, en el caso de la visión estarían formadas por todos aquellos elementos que se encuentran dentro de su área de visión clara (Sección 4.3.1.2), es decir, lo que ve frente a él, y para el caso de la audición seria todo sonido en un radio cercano al agente.

Las percepciones indirectas incluyen el resto de la información que llega al agente, y que no se encuentran dentro de su campo de percepción directa; ejemplo de estas percepciones son, en el caso de la visión, las que se ubican dentro de su área de visión difusa, o en el caso de la audición, los sonidos que se originan lejos del agente, pero dentro de su radio de audición.

En la Sección 5.1.1, sobre el protocolo de comunicación, se presentaron los campos que componen el mensaje de percepción que llega al SCC, el cual consta básicamente de la información relacionada a lo percibido y su tipo (si es directa o indirecta).

Al recibir las percepciones, el Subsistema Analizador determina cuales son recientes y cuales no lo son, manteniendo intactas en la Zona Flash aquellas que viene percibiendo, y borrando todas las demás. A las nuevas percepciones (que no aparecen en la Zona Flash), el Subsistema Analizador les añade la hora actual en que las recibe, y el impacto que tendrá ésta en su velocidad de aprendizaje, para luego almacenarlas en la Zona Flash. La

Hora permite saber cuanto tiempo lleva el agente teniendo la misma percepción, lo cual es útil, por ejemplo, para no recalcular el camino que seguirá el agente para ir a algún sitio, ya que cada nueva percepción puede representar un obstáculo en su trayectoria, con lo que tendría que replantearse la ruta a seguir.

Como se explicara anteriormente (Sección 5.2.2), mientras una percepción permanece en la Zona Flash, esta va siendo memorizada por la Memoria Neuronal. Uno de los factores que afecta la velocidad de memorización (Sección 5.2.4.1) esta asociada a la propia percepción. Por ejemplo, en un parque los niños fijaran más rápido en su memoria los lugares de juegos que los adultos, ya que para ellos esa percepción es de mayor interés. Esto se logra incorporando a la percepción un nivel de impacto, el cual es determinado por el Subsistema Analizador mediante un esquema de reglas que considera el tipo de la percepción, e información tomada del resto del Motor de Conocimiento (gustos, relaciones de parentesco, amistad, otros elementos relacionados, etc). La Figura 5.6 muestra lo aquí expuesto.

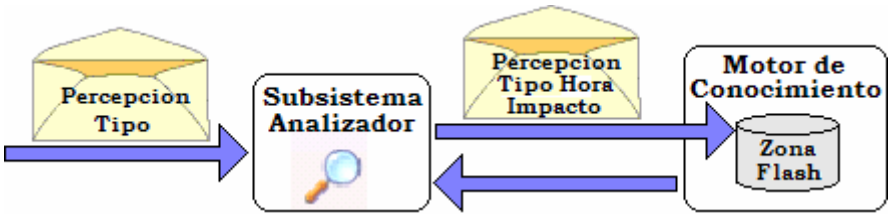

Figura 5.6. Actualización de la Zona Flash

Actualizar Factibilidad de Objetivos

La segunda tarea del Subsistema Analizador es actualizar la factibilidad de cada objetivo; es decir, de establecer la probabilidad que tiene el agente de lograr cada uno de ellos en caso de que se lo proponga.

Todo AVA cuenta con un repertorio limitado de cosas que puede hacer; de hecho, su comportamiento es motivado por el interés de lograr uno de los varios objetivos que se puede auto plantear.

Un comportamiento motivado por objetivos implica que el agente cuenta con un conjunto de ellos, que dictan en todo momento su propósito.

La elección del objetivo que el agente persigue dependerá (de entre otros factores discutidos en detalle en el siguiente capítulo) de la factibilidad del mismo. Por ejemplo, en un parque virtual con solo ruedas para jugar, el objetivo de “jugar en un subibaja” no será factible, por lo que el agente

jamás se planteará jugar en el subibaja, y seguramente opte por hacerlo en la rueda.

Debido a que mucha de la información que maneja el agente en su Motor de Conocimiento incluye incertidumbre, se hace necesario el manejo de grados de certeza para el cálculo de la factibilidad de los objetivos. Para ello, el Subsistema Analizador consta de un Sistema Experto basado en Lógica Difusa, que utiliza la información de lo que percibe (Memoria Flash), de lo que recuerda (Área de Memoria), y de sus gustos (Base de Conocimientos), para determinar el porcentaje en que las condiciones están dadas (según el conocimiento del agente) para el éxito de cada objetivo. Dicho valor es asociado al respectivo objetivo (mantenido en el Motor de Conducta) para ser considerado luego en la toma de decisiones.

Aunque un sistema experto basado en redes bayesianas ([CAST97]) hubiese sido una buena opción para el desarrollo de este subsistema, preferimos usar, por razones prácticas, un esquema de reglas heurísticas apoyadas en lógica difusa ([SILE04]).

5.4 Subsistema de Estados Internos

Para lograr simulaciones realistas de comportamiento de agentes virtuales autónomos, no es suficiente contar con la información procedente de las percepciones y del conocimiento del agente sobre su entorno para establecer sus acciones. Buena parte de las decisiones que toman los seres humanos están influenciadas en un alto porcentaje por su estado anímico. Estados como la alegría o la tristeza afectan la manera en que las decisiones son tomadas; qué información considerar, y de qué manera utilizarla.

La Técnica de Estados Internos emplea el uso de variables, modelos matemáticos, y sistemas de reglas, para emular emociones (tales como la felicidad, el aburrimiento, el miedo, etc) y condiciones físico-mentales (tales como hambre, soledad, energía, etc). Estas variables son tomadas en cuenta para la elección de acciones del agente, incluyendo la expresión corporal del mismo, otorgándole mayor realismo a su desenvolvimiento.

El empleo de esta técnica no es algo nuevo. En [TERZ94, TU96], los autores utilizaban 3 estados internos (hambre, libido, y temor), para determinar el objetivo y las acciones de sus criaturas virtuales. En el caso de los humanos virtuales, podemos encontrar trabajos como [BENCH98, CAIC00] en los que el repertorio de estados internos es un poco más amplio,

llegando a considerar el agrado, el gozo, la admiración, etc. Estos trabajos han sido comentados en el Capítulo 3.

Sin embargo, son los estados internos considerados, los modelos empleados, y la forma en que son utilizados, lo que le otorga originalidad a su empleo.

A diferencia de los trabajos aquí comentados en los que cada estado interno esta estrechamente asociado a un objetivo (representando el deseo de ejecutar dicho objetivo), y en los que por ejemplo, si el agente tiene un libido alto entonces busca aparearse [TERZ94], o si el agente esta alegre entonces baila [CAIC00], consideramos importante resaltar que en este trabajo se busca que los estados internos ejerzan una influencia menos determinante sobre las decisiones, para minimizar los comportamientos deterministas (predecibles). Para ello separamos lo que es *el deseo* o *las ganas* de realizar determinadas acciones (porque las condiciones son favorables), de lo que son las variables físico-emocionales que contribuyen o desfavorecen a dicho deseo.

Esta sección presenta los modelos empleados para emular los estados internos, considerados por el sistema desarrollado para realizar las simulaciones de sus agente virtuales autónomos, y que contribuyen a establecer el nivel de deseo para la selección de un objetivo.

Los estados internos incorporados al sistema desarrollado en esta investigación, son descritos a continuación.

5.4.1 Energía

La función de energía, *E*, mide la fuerza física del agente virtual en cada momento, de donde se puede establecer su nivel de cansancio.

La función se define como,

$$E = \alpha E_1 + (1-\alpha)E_2 \tag{5.1}$$

donde E_1 y E_2 son funciones que representan el cansancio y la recuperación, respectivamente, y $\alpha \in \{0,1\}$ indica si el agente se encuentra realizando una actividad física ($\alpha = 1$) o no ($\alpha = 0$).

Cada vez que el agente ejecuta cualquier clase de actividad, *la función de cansancio*, E_1, varia según la siguiente ecuación,

$$E_1(m,R,t,\rho) = \max\left\{0{,}100 - \rho e^{t\left(\frac{4.6}{mR}\right)}\right\} \tag{5.2}$$

donde $m > 0$ es el máximo número de unidades de tiempo en el que el agente se cansa completamente realizando actividades de esfuerzo normal, $R \in (0,1]$ es la capacidad de resistencia del agente, t es el tiempo transcurrido desde iniciada la actividad, y $\rho \in \{1,1.25\}$ es una variable que describe cuan suave o dura es la actividad que realiza el agente.

Del mismo modo, si el agente se encuentra en un estado de reposo, *la función de recuperación*, E_2, varia según la ecuación,

$$E_2(n,r,t) = \min\left\{100, e^{t\left(\frac{4.6}{n(1-r)}\right)}\right\} \tag{5.3}$$

donde $n > 0$ es el máximo número de unidades de tiempo que un agente necesita para alcanzar su nivel máximo de energía con $r = 0$, $r \in [0,1)$ es la velocidad de recuperación del agente, y t es el tiempo trascurrido desde iniciado el reposo.

Las Ecuaciones 5.1, 5.2, y 5.3, ocasionan el efecto de que cuando el agente virtual este ejecutando alguna actividad, él esté bajo un proceso de agotamiento, pudiendo mantener lo que están haciendo hasta que *E=0*; pero en el momento en que detiene la actividad, su energía comienza a incrementarse según la función de recuperación.

La activación de E_1 por E_2, o viceversa (ya que ambas no pueden estar activadas simultáneamente) implica un cambio en el objetivo del agente. Por ejemplo, cuando la energía es muy alta, $E > E_u$ (donde $E_u < 100$ es un valor umbral superior), el agente probablemente realizara alguna clase de actividad física o intelectual (la elección final dependerá de los deseos del agente, la factibilidad de los objetivos, entre otros). Después de un rato, la energía alcanza un umbral inferior, $E < E_l$, y el agente inmediatamente inicia una búsqueda de un lugar para descansar (quizás una banca para sentarse). El valor E_l es estrictamente mayor que *0* con el propósito de permitirle al agente contar con la suficiente energía para moverse hasta el lugar a descansar. Si la energía disminuye a *0*, no dispondría de energía para hacer nada, y tendría que quedarse quieto in situ para recuperarse lo suficiente como para seguir con sus acciones.

Es importante notar que la variable t en (5.2) y (5.3) es inicializada cada vez que α cambia de valor, de la siguiente manera,

$$t = \frac{1}{4.6}\begin{cases} \log(\max\{E,1\})n(1-r) & \text{Si } \alpha \text{ cambia a } 0 \\ \log\left(\frac{101-E}{\rho}\right)mR & \text{Si } \alpha \text{ cambia a } 1 \end{cases} \quad (5.4)$$

5.4.2 Timidez

La timidez mide las ganas de relacionarse con otros agentes.

La función de timidez viene dada por,

$$T(S,l,t) = \begin{cases} \max\{0,100 - 200\frac{t}{l}S\} & \text{Si el agente esta solo} \\ \min\{100,100\frac{t}{l}(1.5-S\} & \text{Si el agente esta acompañado} \end{cases} \quad (5.5)$$

donde $S \in [0,1]$ es el parámetro de sociabilidad del agente (siendo mayor a medida que se acerca a 1), $l > 0$ es el número de unidades de tiempo que tarda el agente en pasar de querer estar completamente solo (*T=100*) a necesitar estar acompañado (*T=0*), o viceversa, con $S = 0.5$; y finalmente, t es el tiempo transcurrido en uno de los dos estados.

Con este estado interno se persigue que el agente sienta deseos, tanto de estar solo, como de buscar sociabilizar con otros agentes.

El parámetro S establece un rasgo de la personalidad de agente, su grado de sociabilidad. Si el agente es muy sociable ($S \approx 1$), su estado interno de *timidez* aumentara mas lentamente en compañía de otros agentes, que si su rasgo de sociabilidad fuese bajo ($S \approx 0$). Por lo que un agente sociable gusta de pasar más tiempo con otros agentes, que aquellos que son tímidos.

Al igual que sucede con la energía, al pasar de un estado a otro, de solo a acompañado o viceversa, la variable t es inicializada de la siguiente manera,

$$t = \begin{cases} \frac{(100-T)l}{200S} & \text{Si el agente pasa de acompañado a solo} \\ \frac{l*T}{100(1.5-S)} & \text{Si el agente pasa de solo a acompañado} \end{cases} \quad (5.6)$$

5.4.3 Aburrimiento

Este estado interno es utilizado para medir el nivel de desinterés que va experimentando el agente por la actividad que esta realizando, a medida que pasa el tiempo.

La función de aburrimiento viene dada por,

$$B(g_k, n_k, t) = \min\{30, 100 - g_k\} + \frac{100}{n_k} t \qquad (5.7)$$

donde g_k es la variable asociada al nivel de deseo del agente de realizar el objetivo k (detalles sobre esta variable en el próximo capítulo, Sección 6.1), n_k es el número de unidades de tiempo que normalmente tardaría el agente en aburrirse del objetivo k, y t es el tiempo transcurrido de haber comenzado a realizar efectivamente el objetivo k.

Cuando un agente se propone conseguir un objetivo, el nivel de aburrimiento con respecto a dicho objetivo se inicializa a un valor opuesto a las ganas del agente de realizar el objetivo en cuestión, tal como lo expresa el primer termino de la función (5.7), ya que t valdría *0*, porque proponerse el objetivo no implica estarlo realizando (por ejemplo, para jugar en una rueda, hay que buscarla primero). Mientras el objetivo no se haya alcanzado (mientras no este jugando en la rueda, por ejemplo), el aburrimiento permanecerá estático, ya que el valor de t seguirá en *0*. Alcanzado el objetivo, el tiempo comienza a avanzar y el aburrimiento a aumentar; al llegar a su máximo, es el indicativo de que el agente ya no tiene interés en seguir con el objetivo, y lo cambia por otro.

5.4.4 Ansiedad

La función de ansiedad, *A*, es una medida de frustración causada por tratar de ejecutar un objetivo y fallar en el intento. Por ejemplo, si un niño solo tiene como objetivo jugar en un subibaja (para lo que requiere de un compañero), puede optar por sentarse en el subibaja a esperar que otro niño lo vea y quiera jugar con él; pero entre mas tiempo pase solo sin que nadie se acerque a jugar con él, mas ansioso se pondrá, hasta que su paciencia se agote y decida hacer otra cosa (cambiar de objetivo).

Definimos la ansiedad de la siguiente manera,

$$A(t,\sigma_k) = 100\frac{t}{\sigma_k} \tag{5.8}$$

donde t es el tiempo intentando alcanzar el objetivo, y σ_k es el tiempo máximo que podría pasar intentando alcanzar el objetivo k.

5.4.5 Hambre y Sed

Para emular la sensación de hambre de los agentes virtuales se implementó el modelo matemático utilizado en [TERZ94], y el cual fue adaptado en este trabajo tanto para los humanos como para las aves virtuales diseñadas.

La función para el hambre, H, es la siguiente,

$$H(d,a,c) = 100 * \min\left\{1,1 - c\frac{d}{a}\right\} \tag{5.9}$$

donde c es la cantidad ingerida de comida ingerida, a es el grado normal de apetito del agente, y $d = digestion(h-u)$, siendo h la hora actual, u la hora de la ultima comida, y la función *digestión* definida como sigue,

$$digestion(j) = 1 - 0.005j \tag{5.10}$$

Para el caso de la sed se emplea el siguiente método iterativo,

$$E_{i+1} = \begin{cases} \max\{0, E_i - 2\} & \text{Si el agente esta bebiendo agua} \\ \min\left(100, E_i + \dfrac{\delta}{30}\right) & \text{en caso contrario} \end{cases} \tag{5.11}$$

donde δ es la velocidad de deshidratación del agente.

5.4.6 Temor

Este estado interno es solo implementado para el ave virtual, el cual lo alerta de una situación de peligro.

La función temor, *T*, es implementada como sigue,

$$T(p) = \max\{100, 100 - (12p)\} \tag{5.12}$$

donde p es la distancia desde el ave al humano virtual mas cercano.

Las aves virtuales implementadas temen a los humanos virtuales, por lo que cuando un ave percibe que un humano se va acercando, su nivel de temor va aumentando, y al rebasar cierto umbral superior, el objetivo de evasión es seleccionado para huir de una posible situación de peligro.

Y con esto se terminan los estados internos implementados, aunque el diseño del SCC permite que podamos añadir de forma muy sencilla nuevas funciones que representen nuevos estados internos, sin entorpecer el desempeño del sistema; o también modificar las funciones de los estados implementados por otros modelos que se adecuen mejor a los propósitos perseguidos, tal como se corrobora en el capítulo de resultados (Capítulo 8).