Proyecto Final de Carrera

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería Técnica en Informática de Gestión

Proyecto AI-Live : Habilidades y Capacidades en Personajes Virtuales

Autora: Mª Pilar Blanco Martínez

Tutora: Susana Fernández Arregui

Agradecimientos:

A mi tutora Susana Fernández, por ayudarme aplicar los conocimientos adquiridos en clase a este ambicioso proyecto. A Daniel Borrajo, por mirar más allá de mis propuestas.

A todos los alumnos y profesores que hacen y han hecho posible que este proyecto siga adelante. Gracias por aportar vuestro granito de arena.

A mi familia por apoyarme de manera incondicional a lo largo de mi carrera y en especial en estos últimos momentos. A mis compañeros por ayudarme y acompañarme a lo largo de la carrera. A Aarón por estar siempre ahí.

Muchas gracias a todos.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

A lo largo de la historia los juegos han tenido siempre una finalidad como divertir o enseñar. Pero desde hace unas décadas los juegos han conseguido llegar más allá. Poco a poco han ido evolucionando para simular y mejorar las capacidades lógicas o mentales de los humanos en distintas situaciones.

Los juegos de estrategia o de simulación de vida social, tienen un objetivo, unas metas, las cuáles deben ser alcanzadas. Dichas metas hacen que el juego se desarrolle acorde a una línea predefinida. Hace una década salió a la venta uno de los juegos que más impacto ha causado en la sociedad. Este juego fue "Los Sims". Lo novedoso de este juego fue que cada personaje poseía su propia personalidad, era controlado de forma individual, hacía posible que éste aprendiera por sí solo y, según una meta final, el personaje interactuará de forma diferente para llegar a cumplirlo.

Con el paso de los años, "Los Sims" fueron creciendo gracias a expansiones y otras versiones hasta llegar hoy en día a "Los Sims 3". Durante estos años los personajes han podido mejorar sus conocimientos, han llegado a ser desde ladrones, a médicos, pasando por ser actores famosos. Todo esto ha sido gracias a la aplicación de la Inteligencia Artificial.

De igual modo AI-Live ha ido creciendo gracias al aporte de varios alumnos. Desde su primera versión donde se creó una ontología y una estructura de comunicación cliente servidor de bajo nivel, implementada en C, hasta la creación de árboles de tecnologías que hace posible crear objetivos y metas.

Este proyecto de fin de carrera (AI-Live: Habilidades y Capacidades en Personajes Virtuales) consiste en dotar a los actores de habilidades, que son aprendidas de diferentes maneras. Gracias a ello, he introducido un pequeño árbol de tecnología que proporciona al juego un objetivo a alcanzar.

Las habilidades son adquiridas por los actores en dos modos: modo práctico y modo teórico, según las actividades que desarrollen.
En un primer momento, el actor dispone de un número limitado de habilidades que puede desarrollar, pero según vaya progresando se le irán habilitando otras.

Como se ha dicho con anterioridad, AI-Live es un juego de simulación de vida real, creado en el 2006, basado en el juego "Los

Sims" siguiendo un modelo cliente-servidor. El servidor tiene como función asignar los turnos al cliente y actualizar el estado de los actores y de las entidades para enviárselo al cliente según las acciones que hayan solicitado. Existe un motor emocional encargado de controlar los gustos, las relaciones con otros clientes, las emociones y, en esta última versión, las habilidades.

En estos momentos existen varios tipos de clientes:

- El cliente GUI: es el encargado de mostrar la interfaz gráfica. En este caso no ha sido modificado.
- El cliente Manual: es un cliente que está controlado de forma manual por el usuario. El usuario se encarga de elegir la acción que va a ejecutar el cliente.
- El cliente Prodigy: utiliza un planificador de tareas para decidir la acción a ejecutar.
- El cliente CLIPS: es el encargado de decidir qué acción se va a realizar, basándose en un sistema de reglas.

1.2 Cometido del proyecto

AI-Live al ser un juego de simulación de vida social, los actores deberían de comportarse de forma similar a la de los seres humanos. Un posible concepto clave para esto, sería el aprendizaje del conocimiento.
El cometido de este Proyecto de fin de carrera es aportar al proyecto AI-Live un poco más de realismo y similitud al juego con la vida real, y con el juego.

En versiones anteriores, los actores eran capaces de comunicarse unos con los otros, realizar acciones según sus gustos, transmitir esos gustos, trabajar, comprar...

Este proyecto se centra en la simulación de personajes que sean capaces de aprender diferentes habilidades.

La vía que abre el aprendizaje es muy extensa desde aprender los conocimientos a ser aplicados en la realización de diferentes acciones, llegar a cumplir objetivos, desarrollar una profesión laboral...

1.3 Estructura del documento

El documento seguirá la siguiente estructura:

- **Apartado 1, Introducción**: en este apartado se expone una pequeña introducción al proyecto, su cometido, y la estructura que va a seguir la memoria.

- **Apartado 2, Estado del arte:** en este apartado se realizará un recorrido sobre la evolución que se ha ido produciendo a lo largo de la historia de los videojuegos y cómo la Inteligencia Artificial ha contribuido a ello. Se expondrán las técnicas usadas y varios ejemplos de cómo los videojuegos las utilizan.

- **Apartado 3**, **Objetivos del proyecto fin de carrera**: se definen las metas que se desean conseguir con la realización de este Proyecto de fin de carrera.

- **Apartado 4, Memoria del trabajo realizado**: se expone el trabajo realizado. Se detalla la arquitectura de la aplicación, el modelo de conocimiento y el modelo de aprendizaje junto al árbol tecnológico, el diseño experimental elaborado y documenta otras mejoras implementadas.

- **Apartado 5, Pruebas y resultados**: se describen las pruebas realizadas, se muestran los resultados. Se muestra un estudio sobre que habilidades son las preferidas de los actores, los intervalos de tiempo que tardan en completarlas y qué modos prefieren.

- **Apartado 6, Presupuesto**: se expondrá los gastos que ha tenido el desarrollo de este proyecto.

- **Apartado 7, Conclusiones del trabajo realizado**: se resumen los avances conseguidos en el proyecto. Se exponen algunos de los problemas surgidos durante el proceso y su posterior resolución.

- **Apartado 8, Líneas futuras de trabajo**: se plantean posibles ampliaciones del sistema y mejoras.

- **Apartado 9, Bibliografía**: se recogen todas las fuentes de información utilizadas.

- **Apartado 10, Anexos:** se recogen todos los documentos que han sido necesarios para la elaboración del proyecto.

2. Estado del arte

2.1 Introducción

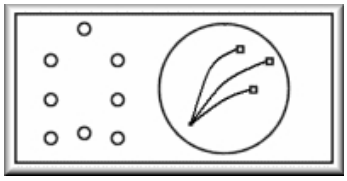
El propósito de este apartado es realizar un recorrido de cómo han ido evolucionado los videojuegos a lo largo de la historia, tanto en aspectos técnicos de software cómo de las máquinas que los han soportado.

El objetivo del estudio del estado del arte es indagar sobre cómo la Inteligencia Artificial ha influenciado en la evolución de los videojuegos y como sus técnicas han hecho que se convierta hoy en día en algo indispensable para su creación.

2.2 Historia de los primeros videojuegos

El camino del videojuego no ha sido tarea fácil. Desde sus comienzos alrededor de la década de los 40 cuando se crearon las primeras súper computadoras, pasando por el boom de los años 80 hasta llegar a lo que hoy denominamos videojuego.

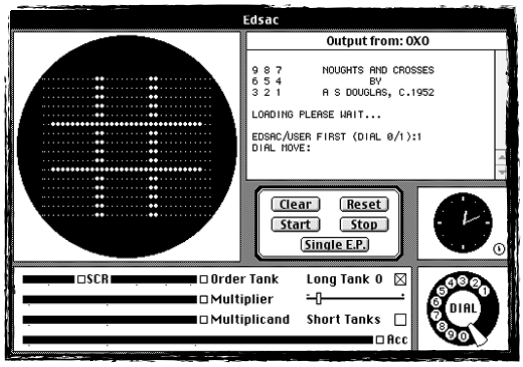
A finales de los años 40, exactamente en 1947, apareció lo que ya podíamos denominar el inicio de los videojuegos. No se le puede llamar en sí videojuego como tal ya que no existía movimiento pero sí se trataba de un experimento real con un dispositivo electrónico de simulación. A este experimento se le llamó “Lanzamiento de Misiles” y fue diseñado por Thomas T. Goldsmith y Estle Ray Mann. En este periodo de la historia la evolución de la tecnología estaba totalmente ligada al mundo militar, por lo que no es de extrañar que esté experimento se tratase de una adaptación de un radar de misiles.



1. Diagrama: lanzamiento de misiles 1947

Debió de pasar media década para que apareciera el que podríamos denominar como el primer videojuego de la historia. En 1952 un estudiante de la Universidad de Cambridge llamado Alexander S. Douglas programó la versión electrónica del juego “Tres en línea”, lo llamó “OXO”, para ilustrar su tesis de doctorado sobre la interacción del ser humano y una computadora.

Existen algunos debates abiertos sobre si en realidad “OXO” fue el primer videojuego de la historia ya que aun no contaba con vídeos animados, pero sí contaba con un modo gráfico y una interacción de juego que su antecesor “Lanzamiento de Misiles” no contaba.



2. Interfaz “OXO”

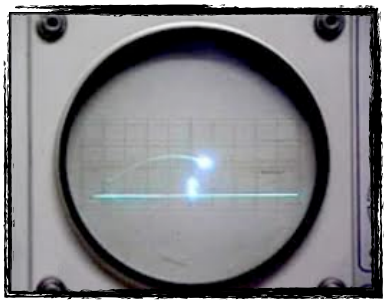
“OXO” fue creado para funcionar en uno de los primeros ordenadores de la historia: EDSAC[1]. “OXO” tomaba sus decisiones en función de los movimientos que el jugador transmitía a la máquina a través de un dial telefónico integrado en el sistema. El juego en sí no tuvo repercusión social ya estaba diseñado para EDSAC y este era único.

En 1958 un físico estadounidense llamado William Higinbothan director de la División de Instrumentación del Laboratorio Nacional Brookheaven, decidió adaptar su tecnología y hacerla más vistosa para amenizar las visitas de puertas abiertas al publico, así nació “Tennis for Two”.

3. Osciloscopio en el que se ejecutaba el juego “Tennis for Two”

[1] Electronic Delay Storage Automatic Computer, fue conectada por primera vez el 6 de mayo de 1949.

William Higinbothan cometió el error de no patentar “Tennis for Two”, pero tanto fue su éxito en las visitas que en 15 años después la empresa “Atari” copiase su idea y creara el que muchos creen como el primer videojuego de la historia “Pong”.



4. Pantalla del juego “Tennis for Two”

En la década de los sesenta y más concretamente en 1961 vio la luz el primer embrión de los juegos galácticos. “Spacewar” fue programado por tres estudiantes , Martin Graetz, Stephen Russell y Wayne Wiitanen, del Instituto Tecnológico de Massachusetts para la computadora PDP-1[2].

5. Pantalla del juego “Spacewar”

“SpaceWar” fue todo un adelanto para la época debido a que contaba, al igual que “Tennis for Two”, con la interacción de dos jugadores pero con un gran adelanto en gráficos ya que estaba formado por un espacio lleno de estrellas con dos naves en movimiento, las cuales podían realizar disparos y moverse en el escenario. El objetivo era eliminar a la otra nave pero sin caer en el centro de gravedad del espacio ya que destruía la nave. El éxito fue tal que el videojuego se extendió por todas las universidades que contaban con computadoras, fuera del ámbito educativo el juego era totalmente desconocido.

---

[2] [2] Minicomputadora desarrollada por la compañía DEC en 1960.

2.3 La Inteligencia Artificial en los videojuegos

Según la Real Academia de la Lengua Española, la Inteligencia Artificial se define como “Desarrollo y utilización de ordenadores con los que se intenta reproducir los procesos de la inteligencia humana”.

Existen muchos tipos de juegos en los que la Inteligencia Artificial aporta su granito de arena. El juego en el que se ha centrado este proyecto de fin de carrera pertenece a los denominados “Simulación de vida virtual”. Antes de que este tipo de juegos irrumpieran en la sociedad los científicos han intentado resolver juegos de lógica como el ajedrez y las damas.

Con el paso de los años han ido evolucionado de tal manera que hemos llegado a niveles a los que la actual industria del videojuego aun no esta preparada.

A continuación explicare de forma breve los comienzos en la resolución de juegos de lógica y los futuros pasos en la industria del videojuego.

2.3.1 Primeros pasos

Los primeros pasos nunca han sido fáciles y menos en este campo en el que existe un estrecha relación con los avances en hardware. A continuación se exponen dos casos de los juegos de lógica más famosos.

Unos de los juegos estrellas siempre ha sido el juego del ajedrez. Desde los inicios los científicos han buscado la manera de automatizarlo.

El primer artefacto autómata conocido fue “El Ajedrecista”, construido por Leonardo Torres Quevedo en 1912, y presentado en la Feria de París en 1914. “El Ajedrecista” simulaba los movimientos de un humano utilizando electroimanes. Debido a la simplicidad del algoritmo que utilizaba no aseguraba la victoria en un número mínimo de movimientos, pero siempre lograba vencer a su rival. Pero no sería hasta 1997 cuando el ordenador “Deep Blue” ganase por primera vez a un experto ajedrecista, Gary Kasparov.

6. "El Ajedrecista" *7. "Deep Blue" contra Gary Kasparov*

Las primeras muestras de Inteligencia Artificial aplicada a la resolución de un juego fue en 1956 cuando Arthur Samuel diseñó y programó un programa que era capaz de jugar a las damas. Este programa guardaba información sobre las diferentes partidas jugadas para poder evitar errores en las futuras partidas. Con este comportamiento podemos decir que el programa llegó ha aprender de tal manera que en un par de años llegó a ganar sistemáticamente a su inventor.

2.3.2 Futuros pasos: Proyecto Natal, Meet a Milo

El proyecto Natal ahora más conocido como "Kinect" es lo último en tecnología para videojuegos. Este proyecto dirigido por la compañía Microsoft bajo las manos de Alex Kipman es un ambicioso proyecto no sólo en hardware si no también la posibilidad del desarrollo de videojuegos con una fuerte base de Inteligencia Artificial.

Este es el caso de "Meet Milo" la evolución de las mascotas virtuales. "Meet Milo" es un videojuego en el que el protagonista es un niño virtual de 8 años llamado Milo con el cual se puede interactuar gracias a la tecnología "Kinect".

8. Capturas del video de presentación de "Meet Milo"

Milo es capaz de mantener una conversación con un jugador de forma totalmente coherente. Es capaz de reconocernos emociones, caras, gestos y diferentes tonos de voz. Esas emociones se ven reflejadas en la cara de Milo. Esta tecnología llega tan lejos hasta el punto en el que Milo es capaz de llegar a reconocer dibujos realizados por los jugadores e interactuar con ellos.

El proyecto en estos momentos se encuentra parado ya que el proyecto es muy ambicioso para la actual industria del videojuego. Pero esto no significa que en un futuro cercano consigan sacarlo al mercado.

2.4 Técnicas de la Inteligencia Artificial en los videojuegos

Unas de las habilidades importantes que tiene el ser humano es la capacidad de resolución de problemas. Está capacidad se está intentando proyectar a los programas a través de diferentes técnicas de Inteligencia Artificial.

La resolución de problemas en diferentes situaciones se pueden aplicar a la resolución de decisiones en los videojuegos haciendo que sean totalmente reales. De ahí que este campo se encuentre tan en auge, pero antes de empezar a nombrar y explicar las diferentes técnicas utilizadas en la programación de videojuegos deberemos de explicar lo que entendemos en este campo por problema.

Podemos definir un problema como una serie de elementos ya sean metas, estado finales u objetivos que debe de cumplir el jugador. La resolución de estos problemas podrían ser un número limitado de movimientos para llegar del estado inicial al estado final.

A continuación explicaré qué técnicas se usan para la resolución de estos u otros problemas en el mundo de los videojuegos.

2.4.1 Pasos a seguir

A la hora de enfrentarnos a un problema en el ámbito de los videojuegos nos solemos referir a vencer a nuestro adversario. Ante esta situación intentamos predecir, superar o adelantarnos a sus movimientos. Para ello se deberán generar todas las posibilidades existentes para conseguir la solución óptima.

Este proceso ideal consiste en generar un árbol de búsqueda con todas las posibilidades pero no siempre podrá realizarse debido a que contamos con un número limitado de recurso, por lo que se realizan otro tipo de búsquedas.

A continuación resumiremos las diferentes técnicas de búsqueda que se utilizan para la resolución de este tipo de problemas según el tipo de juego al que nos enfrentamos.

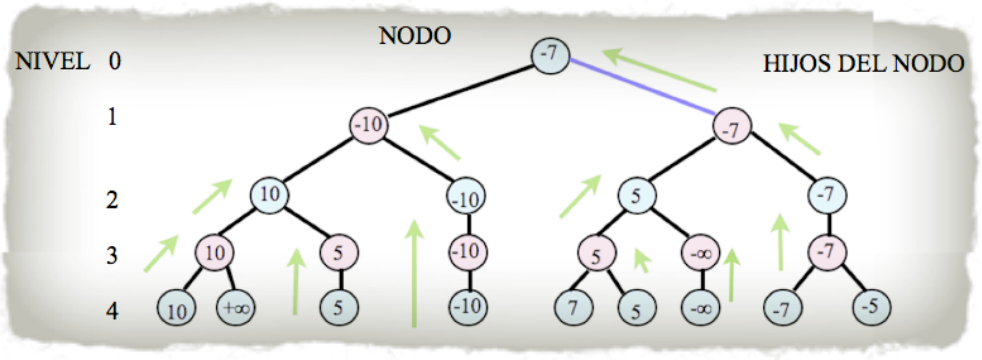
2.4.1.1 Algoritmos utilizados en juegos con adversarios

Los algoritmos utilizados para juegos multijugador o con los que nos enfrentamos a otros adversarios utilizan los típicos algoritmos de búsqueda. A continuación explicaré los diferentes algoritmos.

2.4.1.1.1 MiniMax

Este algoritmo es utilizado en la teoría de juegos para minimizar la pérdida máxima esperada con una información completa ya que cada jugador conoce el estado de su adversario. El método utilizado para la resolución de este algoritmo es el cálculo recursivo.

La idea principal con el que parte el algoritmo es la elección del mejor movimiento para cada jugador siempre y cuando el adversario escoja el peor.



9. Ejemplo del árbol generado por el algoritmo MiniMax

Los componentes que forman el árbol que se generará serán los siguiente:

1. **Nodos:** Representa la situación actual del juego.
2. **Hijos del nodo:** Son las posibles situaciones a las que se puede acceder según el tipo de movimiento que realicemos
3. **Nivel:** Contiene todas las posibles situaciones para cada jugador.

Los pasos a seguir para generar el algoritmo son los siguientes:

1. Se genera el árbol del juego a partir de la situación actual y con una profundidad determinada según el número de interacciones deseadas.

2. Se calcula el valor de la función de utilidad para cada nodo terminal del árbol. Para cada nodo tendremos en cuenta la situación en la que nos encontramos si es beneficiosa será un MAX y si es perjudicial es un MIN.

3. Se calcula valor de los nodos superiores a partir del calor de los nodos inferiores calculados en el anterior paso. Según vemos en el ejemplo mostrado anteriormente vemos como suben los valores más beneficiosos al nodo superior.

4. Una vez se han ido rellenando los niveles superiores llegaremos al nodo de nivel 0 donde se encuentra el valor que escogeremos para la jugada.

El algoritmo MiniMax explora todo los nodos del árbol asignando un valor a cada nodo mediante una función de utilidad, empezando siempre por los nodos de profundidad n hasta llegar al nodo de nivel 0.

El algoritmo de búsqueda MiniMax es utilizado para juegos en los que tenemos un adversario, es decir para aquellos juegos con 2 componentes en los que los jugadores tendrán turnos alternos y del que disponemos en todo momento de la información total de nuestro adversario.

Este algoritmo se caracteriza por identificar a cada jugador con un estado, es decir, uno corresponderá al Max y otro al Min. Esto significa que el jugador que comience la partida se le asignará el estado Max, normalmente seremos nosotros, cuyo objetivo será encontrar los movimientos que nos permitan ganar a nuestro adversario independientemente de los movimientos que realice el jugador MIN, en este caso nuestro adversario.

Un concepto que debemos de explicar es la función de evaluación heurística que consiste en una función que calcula los valores más elevados para indicar la situación más favorable en el caso contrario valores negativos para indicar la situación más favorable para nuestro adversario.

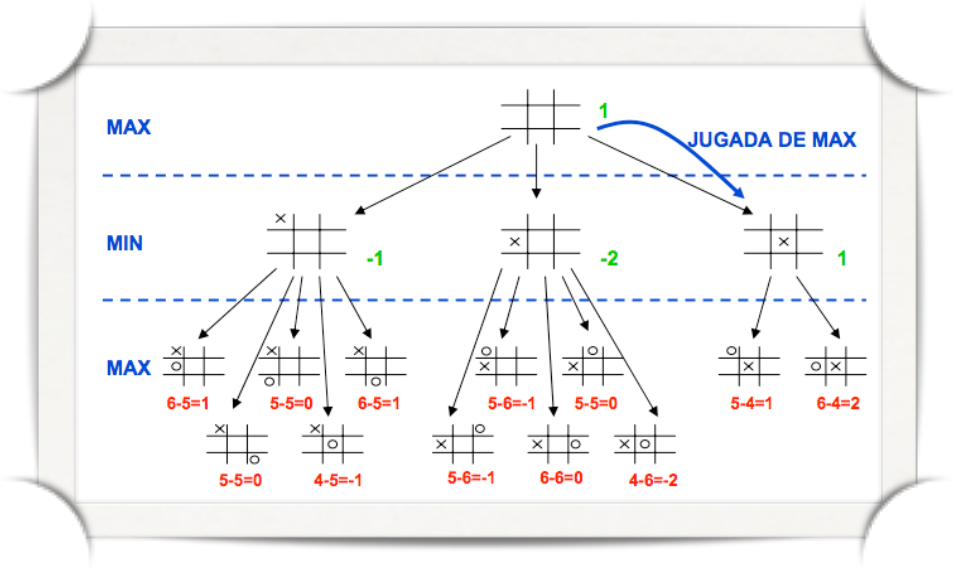
A continuación se ilustra el algoritmo adaptándolo a la resolución del juego OXO, más conocido como “Las tres en raya”.

Para resolver este juego utilizaremos la siguiente heurística:

$$E(n)=X(n) - O(n)$$

donde:

- E(n) es la función.
- X(n) son el número de movimientos posibles para el Max, en este caso el jugador de las X.
- O(x) son el número de movimientos posibles para el Min, en este caso el jugador de las O.



10. Esquema MiniMax juego OXO sacado de los apuntes de Inteligencia Artificial Universidad de Valladolid 2007

Una de las evoluciones posible del algoritmo MiniMax, es aplicarlo dependiendo del adversario al que nos enfrentemos. Al contrario del MiniMax, el algoritmo decidirá la hora de explorar el árbol del juego, que nuestro adversario elegir siempre el camino óptimo.

Esto puede convertirse en una arma de doble filo, ya que dependiendo de la inteligencia de nuestro adversario el movimiento que pensamos nosotros que será el óptimo puede que no lo sea para él y siga otra estrategia.

Por lo que este tipo de algoritmo deberá llevar consigo técnicas que le permitan aprender y así poder ir almacenando conocimiento sobre su adversario a lo largo de la partida, para poder establecer y adelantarse a los movimientos de su oponente a lo largo de la partida o juego.

2.4.1.1.1.1 Mejoras del algoritmo MiniMax

En el apartado anterior he descrito como funciona el algoritmo de búsqueda MiniMax, pero existen unas mejoras que se le aplican al algoritmo para ser más optimo.

2.4.1.1.1.1.1 Método de poda Alfa-Beta

Este método mejora la eficiencia del algoritmo de búsqueda MiniMax que he explicado en el anterior apartado para casos en los que la ramificación es muy elevada y por consiguiente su profundidad deberá ser recortada.

Las mejoras que incluye es la necesidad de hacer dos pasadas al árbol de búsqueda, una para generarlo y otra pa realizar una evaluación y decidir que valores son los mejores e ir descartando aquellas ramas en las que los valores no sean favorables.

Este método aporta dos nuevos componentes para representar los umbrales:

- Alfa (α) : representa el valor de la cota inferior que se le asigna en último término a un nodo máximo. Esto significa que se le asigna el máximo de los valores encontrados en los nodos sucesores.

- Beta (B) : representa el valor valor de la cota superior que se le asigna en último término a un nodo mínimo. Esto significa que se le asigna el mínimo de los valores encontrados en los nodos sucesores.

De esta forma el método puede acotar los valores en la búsqueda del mejor valor para nuestro movimiento.

Las reglas de finalización depende del nodo en el que nos encontremos:

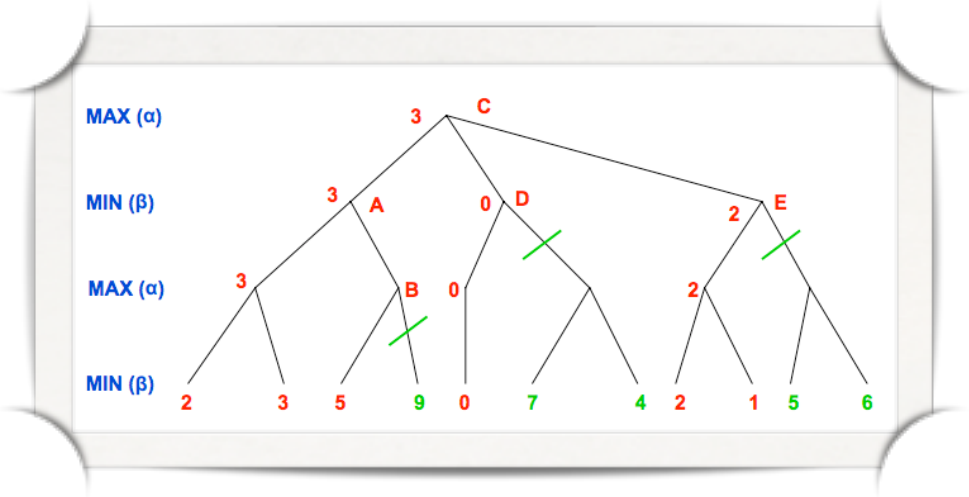
Para nodos máximos la condición aplicada para la poda será aquella en la que αp >= B(p-1), esto significa que podremos podar una rama siempre y cuando un nodo máximo tenga un valor alfa sea mayor o igual que su nodo antecesor beta.

Para los nodos mínimos la condición aplicada para la poda será aquella en la que α(p-1) sea >= Bp, esto significa que podremos podar una rama siempre y cuando un nodo mínimo tenga un valor alfa por encima del valor beta de su antecesor.

Pero ¿cómo sabemos que camino debemos descartar y con cuál nos debemos de quedar? Para poder responder a esta preguntar explicaré con un ejemplo las reglas que se utilizan.

Los pasos a seguir son:

- Buscamos en profundidad hasta llegar al nivel de profundidad máximo.
- Aplicamos la función heurística al nodo y a todos los de su mismo nivel.
- Asignamos los valores al padre.
- Este valor lo ofertamos a su antecesor.
- Descendemos por otra rama para explorar al resto de terminales hasta encontrar los valores de parada.



11. Ejemplo poda alfa-beta sacado de los apuntes de Inteligencia Artificial Universidad de Valladolid 2007

Según se observa en el ejemplo:

- El nodo A tiene un valor B=3 esto significa que A no podrá valer más que 3.
- El nodo B se poda porque su valor más pequeño es de 5 Max 5>3, así que queda descartado.
- El valor del nodo C tiene un valor de α=3. esto significa que no tendrá un valor más bajo que 3 por ser un nodo Max.
- El nodo D es podado ya que en este nodo Min es 0<3.
- El nodo E es podado ya que en este nodo Min es 2<3.

2.4.1.1.1.1.1.1 Mejora del poda Alfa-Beta

El método llamado Aspiration search es una mejora de la búsqueda de Poda alfa-beta.

Esta caracterizado por realizar una búsqueda mejorada añadiendo aproximaciones al resultado esperado para las cotas alfa y beta haciendo así una estimación de la posible desviación que podemos encontrarnos en un valor.

Los resultados que obtendremos al realizar esta búsqueda serán más reducidos proporcionándonos así un número menor de nodos. El único inconveniente que aparece al utilizar este algoritmo es el posible fallo en la búsqueda si no nos encontramos en un caso dentro de nuestras expectativas, esto conllevará realizar una nueva búsqueda.

Para poder realizar las búsquedas contamos con una ventana *"ventana de aspiración"* en la que se calcularán los valores de la posible desviación que puede tener el valor a analizar. Gracias a esta ventana podremos realizar un menor número de búsquedas en los nodos ya que contaremos con unos limites.

2.4.1.1.1.1.2 Algoritmo NegaMax

Esté método es la versión más compacta del algoritmo MiniMax, simplemente se basa en utilizar una búsqueda para sacar los valores negativos de nuestro adversario en vez de realizar una búsqueda para sacar los valores positivos y negativos según se trate de nuestro movimiento o del movimiento del adversario.

2.4.1.2 Algoritmos utilizados en juegos sin adversarios

Para los juegos sin adversarios o también conocidos unipersonales como por ejemplo el 8-puzzle,el Mahjong , el buscaminas o los solitarios de cartas se suelen aplicar algoritmos de búsqueda heurísticas muchos mas sencillos que los explicados en el anterior apartado.

*2.4.1.2.1 Algoritmo A**

El algoritmo A* es un algoritmo heurístico que implementa la búsqueda best-first, primero el mejor. Es uno de los algoritmos más utilizados en la resolución de juegos sin adversario. Uno de los primeros juegos que lo implementaron fue Pac-Man, para el movimiento de los fantasmas.

El algoritmo tiene como meta encontrar los movimientos necesarios para llegar de un estado inicial dado al estado final, eligiendo siempre el camino más favorable. Esto nos da la posibilidad de optimizar el número de movimientos para poder completar el recorrido con el menor número posible.

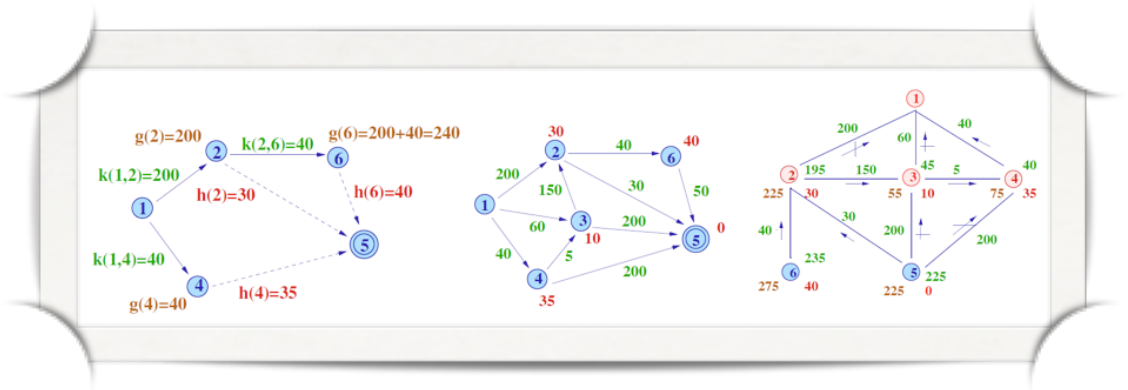
Para poder realizar este cálculo el algoritmo contará con una función de evaluación que irá seleccionando los nodos de la red. Para el caso de querer encontrar la distancia más corta desde un nodo origen a un nodo destino podemos aplicar esta función:

$$f(n)= g(n)+h(n)$$

donde:

- f(n) es la función de evaluación.
- g(n) es la función de coste entre el nodo origen al nodo n.
- h(n) es la función heurística que mide la distancia entre el nodo n al nodo final.

El siguiente ejemplo ilustra el proceder del algoritmo A*:



12. Ejemplo A sacado de los apuntes de Inteligencia Artificial Grupo PLG UC3M IA 2008*

2.5 Juegos de simulación de vida

Los juegos de simulación de vida pertenecen al genero de los videojuegos de simulación. Este tipo de juegos también son conocidos como videojuegos de vida artificial en el que el jugador controla las acciones de un personaje que imita el comportamiento de los seres vivos.

Los juegos que simulan la vida real se encuentran en auge desde que el uso de internet llegó al ámbito domestico. Las redes sociales han aprovechado este tirón para crear numerosos portales en los que se interactúa con personas de diferentes partes del mundo a través de avatares. A continuación explicaremos algunos de estos juegos.

2.5.1 Historia de "Los Sims"

El juego de simulación de vida más famoso de todos los tiempos es "Los Sims". Fue creado en la década de los años 90 por Will Wright fundador de la compañía "Maxis Studio", actualmente conocida como "Maxis Software" después de ser comprada por EA (Electronic Arts) en 1997. El nombre elegido para el juego fue "Los Sims", el nombre está inspirado en la palabra "Simulator".

La idea de crear un juego que simulase la vida de cualquier ciudadano vino de una experiencia vivida por Will Wright. A principios de la década de los años 90 Will Wright tuvo un desafortunado suceso, sufrió un incendio que devasto su hogar. Will Wright se vio obligado a rehacer su casa y comenzar una nueva vida. En ese momento se dio cuenta de los pasos que tuvo que realizar para poder volver a la normalidad.

Este suceso le marco de tal manera que decidió plasmar dicha experiencia y en 1993 propuso a "Maxis Studio" la idea de crear un videojuego en el que su objetivo fuese poder crear una vida y la toma de decisiones marcarán un objetivo diferente para cada jugador sin tener un final predeterminado. Esta idea gusto tanto que el juego se desarrollo en pocos años terminando a finales de los 90, viendo la luz en febrero del 2000.

13."Los sims" *14. Will Wright con una de sus creaciones* *15. 1º Expansión*

Will Wright es considerado uno de los hombres más influyentes en la historia de los videojuegos. En 2001 le fue concedido en la feria Game Developers Choice Awards el premio a todo una vida de logros y en 2002 se convirtió en el primer miembro del Salón de la Fama de La Academia de las Ciencias y las Artes Interactiva [3].

2.5.2 Los Sims Social

Con le éxito de las redes sociales la compañía EA (Electronic Arts) decidió realizar una importante inversión con diferentes juegos adaptándolos a las nuevas tecnologías.

"Los Sims Social" es la versión social del famoso juego "Los Sims" que vio la luz el 8 de agosto de 2011. La plataforma elegida por EA (Electronic Arts) para lanzar el juego fue Facebook, debido a que es una de las redes sociales más famosas del momento. "Los Sims Social" se ubico en el top 10 de los juegos más populares de Facebook el mismo día de su lanzamiento [3].

Las diferencias existentes entre esta versión y la versión clásica del juego es la necesidad de interactuar con diferentes personas en modo online para poder conseguir sus objetivos.

2.5.3 Black&White

"Black&White" es un videojuego para plataforma PC creado por Lionhead Studios para la compañía EA (Electronic Arts).

"Black&White" tiene como objetivo la difícil tarea en la que el jugador deberá actuar como un dios. El jugador deberá decidir que tipo de dios será bueno, malo o neutral según las acciones que tome.

---

[3] Información sacada del punto [Vadejuegos] de la bibliografía.

El objetivo es una ardua tarea ya que las decisiones deberán de ser tomadas a tiempo real.

El juego se desarrolla en el mundo "Black&White" a través de una mano flotante con la que el jugador podrá interactuar con los diferentes elementos del mundo. El mundo consta de diferentes islas que están formadas por un conjunto de aldeas con sus aldeanos y una criatura gigante.

El objetivo del juego es la dominación de las aldeas y conseguir que sus aldeanos se conviertan en creyentes. Para poder llevar esta tarea a cabo contaremos con la presencia de una criatura gigante que nos permitirá llevar acabo nuestro cometido y ayudarnos en la lucha con otros dioses. Esta criatura es enseñada por el jugador para que realice ciertas tareas y mantenga a los aldeanos y a las otras criaturas en su linea.

16. Capturas del juego "Black&White"

La base de Inteligencia Artificial del juego está en la llamada criatura. Al inicio del juego deberemos de elegir entre tres tipos diferentes: el tigre, la vaca y el mono. A lo largo del juego deberemos ir enseñándola para que trate a los humanos según el bando elegido.

Para ello le enseñaremos diferentes hechizos y acciones como recoger piedras, llenar graneros, cometer incendios, comer, descansar... y la criatura irá aprendiendo a realizarlos según las necesidades y situaciones en la que los aldeanos se encuentren. Según la criatura elegida el aprendizaje será más lento o rápido. Es un juego muy completo con muchas posibilidades de jugabilidad y con una base de Inteligencia Artificial muy interesante.

2.5.4 Fable

"Fable" es un juego desarrollado por "Lionhead Studios" para la compañía "Microsoft Game Studios". La primera versión fue lanzada para consola pero tras su éxito se creo una versión más avanzada para PC.

"Fable" es un videojuego caracterizado en la época medieval cuyo protagonista es un niño el cuál deberá vengar la muerte de su familia. El ejemplo práctico de la Inteligencia Artificial en este juego se basa en el desarrollo del protagonista y del mundo que le rodea que se encuentra en constante cambio. El niño irá creciendo y moldeándose según las acciones tomadas por el jugador hasta convertirse en un hombre capaz de llevar a cabo su venganza.

17. "Fable": ejemplo de las posibles evoluciones del personaje.

El personaje podrá evolucionar como un héroe o podrá llegar a convertirse en un villano, todo depende del camino elegido. Todas las acciones y decisiones tomadas harán que evolucione en un sentido u otro, en el juego no existe el azar, si no que todo está conectado y tiene sus consecuencias, tanto es así que pequeños gestos como las horas que se encuentre el personaje expuesto al sol, la alimentación que lleve o el simple hecho de enamorarse harán que el personaje evolucione de forma diferente.

2.5.5 Fritz

"Fritz" es uno de los mas famosos juegos de ajedrez de estos días. Fue diseñado y programado por los alemanes Franz Morsch y Mathias Feist. "Fritz" es un juego de ajedrez que se empezó a diseñar a principios de los años 90's tras fijarse la compañía Chessbase en Morsch y Feist los cuales habían realizado un proyecto similar llamado "Knigtstalker".

"Fritz" se hizo famoso en la competición de 1995 realizada en Hong Kong donde se ponía a prueba varios programas de ajedrez. "Fritz" causo tanto interés que se convirtió en el software más rentable de la compañía Chessbase.

A finales de los años 90, más exactamente en 1998, “Fritz 5.0” se le categoriza como unos de los programas más fuertes asignándole la puntuación “ELO” de 2460 y en año 2000 consiguió superar la puntuación “ELO” de 2600 en 7 puntos siendo el primer juego programado en romper dicha barrera.

Los méritos de “Fritz” no quedan sólo aquí, si no que como nombramos en el apartado “2.3.1 Primeros pasos”, una de sus versiones mejoradas llamadas “Deep” en las que se aprovecha la tecnología de varios núcleos que presentó el superordenador “Deep Blue”, consiguió ganar por primera vez a un experto ajedrecista, Gary Kasparov.

En la actualidad “Fritz” está retirado del campeonato mundial, pero sigue siendo uno de los juegos de ajedrez más famosos y vendidos de todos los tiempos.

2.6 Conclusiones

Tras haber seguido los pasos de la evolución de los videojuegos en las últimas décadas he podido observar que a partir de sencillas ideas como las de William Higinbothan, para amenizar las visitas en su laboratorio, dieron lugar a lo que fue el germen de este mundo fantástico que hoy en día mueve millones de euros y lo que es más importante ilusiona en el mundo a millones de personas. Las pequeñas ideas siempre llevan tras de sí esfuerzo e ilusión.

En pocos años hemos podido observar grandes cambios en este mundo de fantasía. Desde los primeros videojuegos en los que primaba la diversión en entornos sencillos en los que sólo se podían hacer pequeños movimientos a realizar videojuegos con la máxima realidad posible en entornos dinámicos y con personajes con los que el jugador se ve identificado, tanto por sus circunstancias como por su aspecto.

Todo esto ha sido posible gracias a la pronta evolución que ha sufrido la tecnología en los últimos años, desde la utilización de osciloscopios para la visualización de los juegos o teléfonos para la comunicación entre el dispositivo y el usuario a la utilización de simples movimientos o gestos como los que se utilizan para la tecnología “Kinect”.

El afán de conseguir realismo ha hecho que la Inteligencia Artificial gane protagonismo en este mundo. Aunque muchos lo ignoren, desde el boom de los videojuegos en la década de los 80, la Inteligencia Artificial ha estado presente en juegos como Pac-Man, en el que se encargaba del comportamiento de los fantasmas.

Es cierto que hoy en día esa mínima parte que antes utilizaba se ha visto incrementada en algunos videojuegos que asemejan comportamientos como es el caso de “Los Sims” o en videojuegos en los que la toma de decisiones a lo largo de la partida destinan el desarrollo de los personajes o de su entorno ya que no existe el azar y todo se encuentra en constante cambio como es el caso de “Fable”.

¿Qué le deparará el futuro a esta industria? Esta pregunta se puede resolver sencillamente si pensamos que este mundo de fantasía se verá impregnado de matices de realidad en la que los usuarios interactúen de forma natural con el medio a través de dispositivos que permitan un comportamiento sencillo sin verse obligados a imaginar que se encuentran en ese mundo, si no que de verdad lo sientan. Este camino será posible si la tecnología avanza de la mano de la Inteligencia Artificial para poder conseguir ese realismo.

3. Objetivos del proyecto fin de carrera

El objetivo principal de este Proyecto de Fin de Carrera es dotar al juego AI-Live de un cliente que sea capaz de adquirir unas habilidades a partir de sus capacidades. Con este comportamiento se quiere dar un mayor realismo al juego asemejándolo a la vida real. Los actores realizarán una serie de actividades tales como estudiar, hacer ejercicios, aprender viendo la televisión o comunicarse con otros actores que les proporcionarán unos conocimientos. Estas actividades serán intercaladas con la realización de otro tipo de actividades, como pueden ser las básicas o las de ocio.

El juego AI-Live contaba con el desarrollo de un cliente que realizaba una serie de actividades lógicas que proporcionaban al actor satisfacer sus necesidades según sus gustos o estado de ánimos. Las actividades se desarrollaban según un tiempo de duración para darle más realismo, ya que dependiendo de la acción que desarrollara el actor tardaría un turno o varios, ya que no es lo mismo comer que trabajar o que dormir.

Una de las novedades añadidas al juego AI-Live es la inclusión de un árbol de tecnología que permite crear una meta. El árbol de tecnología proporciona al juego una mayor jugabilidad y hará que los actores puedan desarrollar las diferentes habilidades centrándose en un objetivo. El árbol contará con una serie de niveles que se irán superando según el actor vaya desarrollando las diferentes habilidades que se encuentren ligadas al árbol de tecnología, ya que no todas las habilidades disponibles se encuentran ligadas al árbol de tecnología.

Otras de las novedades es que cada actor contará con unas capacidades especificas con las que se verá influenciado a la hora de poder adquirir conocimiento.

El aprendizaje lo podemos definir como un proceso de cambio relativamente permanente en el comportamiento de una persona generando experiencia (Feldman, 2005). Según una de las 7 leyes que comprenden el aprendizaje, la cual dice: “Ley de la Individualidad: Es aquella que nos indica que el nivel de aprendizaje depende del nivel de capacidad de las personas, ya que todas las personas tienen un nivel de capacidad diferente” (Huacho, 2011). Esto hará que el aprendizaje este influenciado por las capacidades que presente el actor, haciendo que se realice de una manera más rápida si sus capacidades son favorables, o en el caso contrario producirse de una manera más costosa si sus capacidades son menos favorables.

En este proyecto las capacidades de los actores están catalogadas en capacidad práctica y capacidad teórica.

La capacidad práctica hará que el actor pueda realizar actividades que conlleven un aprendizaje práctico como es el caso de realizar ejercicios a través del uso de libros. Por otro lado la capacidad teórica será utiliza el aprendizaje teórico como es el caso de estudiar o intercambiar conocimientos con otros actores.

Gracias a estos aportes el juego AI-Live se ve dotado de una mayor similitud al comportamiento de los seres vivos y lo asemeja aun más a los diferentes juegos de simulación de vida.

4. Memoria del trabajo realizado

4.1 Introducción

En este apartado de la memoria se realiza la descripción del trabajo realizado para el juego AI-Live. El proyecto realizado aporta una nueva funcionalidad al juego AI-Live, el desarrollo de habilidades a partir de las capacidades de un actor.

Al inicio del proyecto el juego AI-Live podría realizar una secuencia de actividades lógicas influenciadas por los gustos o el estado de ánimo del actor. También contaba con la percepción del tiempo vinculado a realizar las diferentes actividades, ya que como pasa en la vida real cada actividad tarda un tiempo en poder ser desarrollada. Este sistema hacía que se asemejara al comportamiento humano en el desarrollo de actividades.

El sistema tenía desarrollado un módulo emocional que permitía la comunicación con otros actores en el cuál se podía realizar diferentes intercambio como gustos, cotillear o simplemente hablar[Jiménez, 2008]. Basándome en esas ideas, decidí adaptar este módulo también a mi nueva funcionalidad, por lo que he implementado la acción de poder intercambiar conocimiento con otros actores. Esto hará que se integren todos los módulos con la nueva funcionalidad.

Para poder desarrollar la nueva funcionalidad he optado por modificar parte del sistema ya existente y crear un nuevo módulo para poder gestionar de una manera más eficiente las actividades relacionadas con las habilidades. Esto conlleva por tanto la creación de nuevos objetos en el escenario, clases y reglas, que serán descritos en los posteriores apartados.

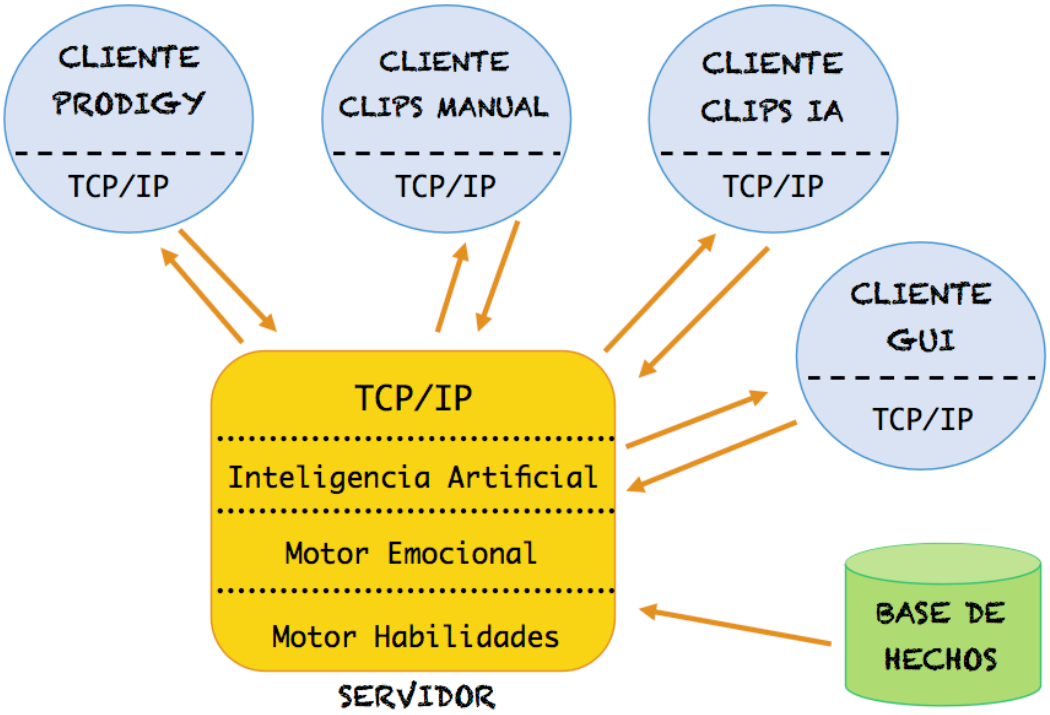
4.2 Arquitectura de la aplicación

El proyecto AI-Live presenta una arquitectura cliente-servidor cuya comunicación se realiza sobre el protocolo TCP/IP. La arquitectura cliente-servidor es un modelo distributivo que reparte las tareas entre los proveedores del servicio, el servidor, y los que solicitan el servicio, el cliente, en nuestro caso los actores. Esta arquitectura reparte la capacidad de proceso entre el servidor y el cliente, en nuestro caso existe más de un cliente.

Las ventajas que presenta este modelo es la centralización del control en los accesos y los recursos. Aunque una de las mayores ventajas que presenta esta arquitectura es la independencia en la

mejorara de cualquier elemento del servidor o/y del cliente, así como su mantenimiento, pudiendo añadir diferentes clientes sin necesidad de modificar el servidor.

El proyecto AI-Live está formado por un servidor y cuatro clientes, (*Cliente GUI, Cliente CLIPS IA, Cliente CLIPS MANUAL, Cliente PRODIGY*) los cuales pueden aumentar en la ejecución del juego ya que el "Cliente CLIPS IA", el cual representa a los actores, podrá ser lanzado tantas veces como actores queramos que tenga el juego.



18. Esquema arquitectura juego AI-Live.

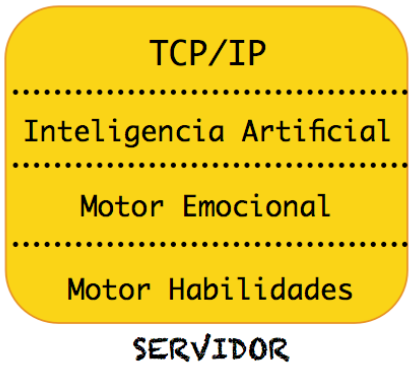
El servidor es el encargado de recibir la acción que quiere realizar cada cliente en su turno. Deberá comprobar que la acción solicitada por el cliente puede llevarse a cabo y deberá actualizar el entorno una vez haya sido realizada para que en el siguiente turno asignado pueda realizarse sin ningún problema.

Los clientes que aparecen en la arquitectura del juego AI-Live son motores de Inteligencia Artificial a excepción del "Cliente GUI" que es el encargado de la interfaz gráfica del juego. Los clientes se encargan de generar peticiones que serán posteriormente tratados por el servidor.

4.2.1 Servidor

El servidor del juego AI-Live es el encargado de establecer los turnos entre los diferentes clientes debido a que gestiona los protocolos de conexión entre los diferentes clientes. El servidor alberga toda la información relevante al estado del universo haciendo posible que ejecute las acciones deseadas por los clientes.

El servidor del juego AI-Live esta diferenciado en cuatro partes:



19. Servidor del juego AI-Live

1) **Módulo principal**: es el encargado de establecer los turnos entre los clientes y las llamas al módulo de Inteligencia Artificial. Se alberga en el fichero server.c.

2) **Módulo Inteligencia Artificial**: es el encargado de ejecutar las acciones realizadas por los clientes clips y contiene el estado del universo . Se alberga en el fichero server.clp

3) **Módulo Motor Emocional**: es el encargado de actualizar el estado de los actores cuando se realizan las acciones de comunicación entre dos o más actores. Este módulo es llamado por el módulo de Inteligencia Artificial. Se alberga en el fichero emotional_engine.clp.

4) **Motor Habilidades**: es el encargado de actualizar el estado de los actores cuando se realizan las acciones de aprendizaje de una habilidad. Este módulo es llamado por el módulo de Inteligencia Artificial. Se alberga en el fichero hability_engine_server.clp.

4.2.2 Clientes

El cliente del juego AI-Live se encarga de realizar las peticiones que son gestionadas por el servido. El juego AI-Live está formado por 4 clientes "Cliente GUI, Cliente CLIPS IA, Cliente CLIPS MANUAL, Cliente PRODIGY" que están gestionados por dos módulos.

1) **Módulo principal**: se encarga de establecer la conexión entre los clientes y el servidor. Se alberga en el fichero client.c.

2) **Módulo ejecución**: se encarga de realizar la ejecución de los diferentes clientes "Cliente GUI, Cliente CLIPS IA, Cliente CLIPS MANUAL, Cliente PRODIGY" del juego AI-Live.

2.1) **Cliente GUI**: es el encargado del modo gráfico del juego representando el universo de realidad simulado en 3D.

2.2) **Cliente PRODIGY**: contiene el motor de Inteligencia Artificial del juego. Realiza la integración del planificar de tareas "PRODIGY" para poder controlar los diferentes actores que se encuentran ejecutando en el juego AI-Live.

2.3) **Cliente CLIPS IA**: contiene el motor de Inteligencia Artificial del juego. Se encarga de la integración del sistema CLIPS basado en reglas que controla las actividades que puede realizar un actor dentro del juego. Este cliente podrá ser lanzado tantas veces como actores queramos tener en el juego.

2.4) **Cliente CLIPS MANUAL:** contiene el motor de Inteligencia Artificial del juego. Se encarga de la integración del sistema CLIPS basado en reglas que controla las actividades que puede realizar un actor dentro del juego. La diferencia con el anterior cliente consiste en que el sistema es manual, por lo que seremos nosotros los que decidamos qué actividades realizará el actor dentro del juego AI-Live a través de una lista de posibles actividades que se genera en cada turno. Este cliente podrá ser lanzado tantas veces como actores queramos tener en el juego.

4.2.3. Protocolo de comunicación

En este apartado de la memoria se explicara el funcionamiento entre el servidor y los clientes del juego AI-Live. Podemos definir un protocolo de comunicación como un conjunto de reglas empleadas para poder realizar un intercambio de información entre los nodos de una red. Para poder realizar el intercambio de mensajes todos los nodos de la red deberán de emplear los mismos protocolos de comunicación. El protocolo de comunicación utilizará diferentes etiquetas como "HOLA,STAG,ACTR,ACTN,STAT,GO,EX"

- **HOLA**: etiqueta utilizada por el cliente y el servidor para iniciar la conexión. La etiqueta lleva diferente información como la versión y las diferentes opciones soportadas por los elementos que intervienen en la comunicación.

- **STAG**: etiqueta utilizada por los clientes para indicar en qué escenario se comenzará el juego.

- **ACTR**: etiqueta utilizada por los clientes para controlar el identificador de cada uno de los actores que se encuentran ejecutando.

- **ACTN**: etiqueta utilizada por los clientes para identificar que acción va a realizar cada actor que se encuentra ejecutado el servidor en el juego.

- **STAT**: etiqueta utilizada por el servidor para comunicar al cliente el estado del juego.

- **GO**: etiqueta utilizada por el servidor y el "Cliente GUI" para comunicar que se ha terminado el módulo actual de la actualización de la información y que otro módulo puede continuar su ejecución.

- **EX**: etiqueta utilizada por el "Cliente GUI" para indicar al servidor que se ha finalizado la ejecución.

Una vez explicadas las diferentes etiquetas utilizas en el protocolo de comunicación se explicará brevemente los diferentes pasos que se realizan por el servidor y los distintos clientes para estableces la comunicación.

4.2.3.1. Pasos Servidor

El servidor realizara cuatro pasos a la hora de establecer la transmisión de información con los clientes. Los pasos a seguir son los siguientes:

1) Inicia el motor de Inteligencia Artificial del servidor.

2) Inicia el analizador de red.

3) Una vez lanzado el cliente debe conectarse a él mediante el protocolo establecido.

4) Realizar el siguiente bucle principal en ejecución:
   - 4.1) Selecciona el cliente que tenga asignado el turno.
   - 4.2) Envía el estado al cliente actual y al cliente GUI.
   - 4.3) Recibe la acción del cliente.
   - 4.4) Envía la acción al servidor de Inteligencia Artificial.
   - 4.5) Recibe el estado de Inteligencia Artificial.

4.2.3.2. Pasos Clientes CLIPS Inteligencia Artificial y Manual

Los clientes "Cliente CLIPS IA, Cliente CLIPS MANUAL" realizaran varios pasos a la hora de establecer la transmisión de información con el servidor. Los pasos a seguir son los siguientes:

1) Realizar la conexión con el servidor.

2) Realizar el siguiente bucle principal en ejecución:
   - 2.1) Recibir el estado.
   - 2.2) Interpretar el estado recibido.
   - 2.3) Elegir la acción a realizar por el actor:
     - 2.3.1) Si es el Cliente CLIPS IA" realizará la elección automáticamente.
     - 2.3.2) Si es el "Cliente CLIPS MANUAL" seremos nosotros los que decidamos qué actividades realizará el actor a través de una lista de posibles actividades que se genera en cada turno.
   - 2.4) Enviar la acción que queremos realizar al Servidor.

4.2.3.3. Pasos Cliente GUI

Los clientes "Cliente GUI" realizará varios pasos a la hora de establecer la transmisión de información con el servidor. Los pasos a seguir son los siguientes:

1) Realizar la conexión con el servidor.

2)Realizar el siguiente bucle principal en ejecución:
- 2.1) Recibir el estado referente a los objetos gráficos.
- 2.2) Representar gráficamente el estado recibido.

4.3 Modelo de conocimiento

El modelo de conocimiento del juego AI-Live está diseñado con el aporte de los diferentes proyectos de fin de carrera que se han ido realizando. Cada proyecto a añadido una serie de clases a la ontología del sistema para poder desarrollar las nuevas funcionalidades.

Al inicio del proyecto la ontología se encontraba albergada en el archivo "ontology.clp", pero poder integrar la nueva funcionalidad y optimizar el mantenimiento he optado por crear otro archivo, "hability_ontology.clp", en el que se encuentran albergadas las nuevas clases creadas para el módulo de habilidad. Sin embargo aquellas clases que ya se encontraban creadas y se han tenido que modificar se han actualizado en el anterior archivo.

A continuación se mostrará de manera gráfica los aportes y los cambios realizados al modelo de conocimiento con la posterior explicación.

4.3.1 Aportes realizados al modelo de conocimiento

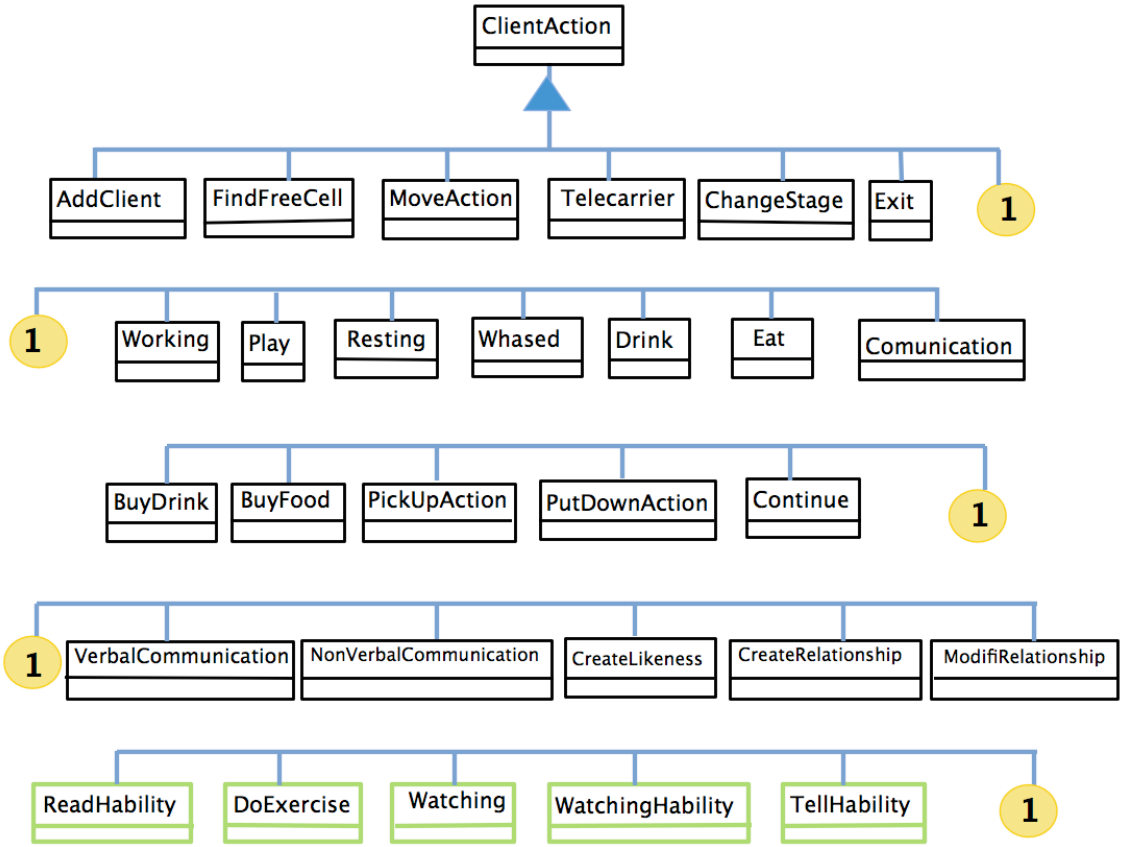
En este apartado se describirá tanto de manera gráfica como escrita los aportes realizados al modelo de conocimiento del juego AI-Live. Primero se realizará la explicación de las clases creadas en el archivo "ontology.clp" y posteriormente la aportación del nuevo archivo "hability_ontology.clp".

4.3.1.1 Aportación en "ontology.clp"

Antes de comenzar con los cambios relacionados con la nueva funcionalidad todas las clases se encontraban dentro de "ontology.clp". Para poder adaptar las necesidades comencé a realizar los diferentes

cambios y aportaciones dentro de este fichero para aquellas clases que serán hijas de las ya creadas.

Las clases añadidas en "ontology.clp" son aquellas que nos permiten realizar las nuevas actividades para el módulo de habilidad. En el siguiente gráfico se pueden observar todas las clases existentes y las creadas para el módulo de habilidad que se representan en color verde.



20. Clases que representan las actividades que puede realizar un actor

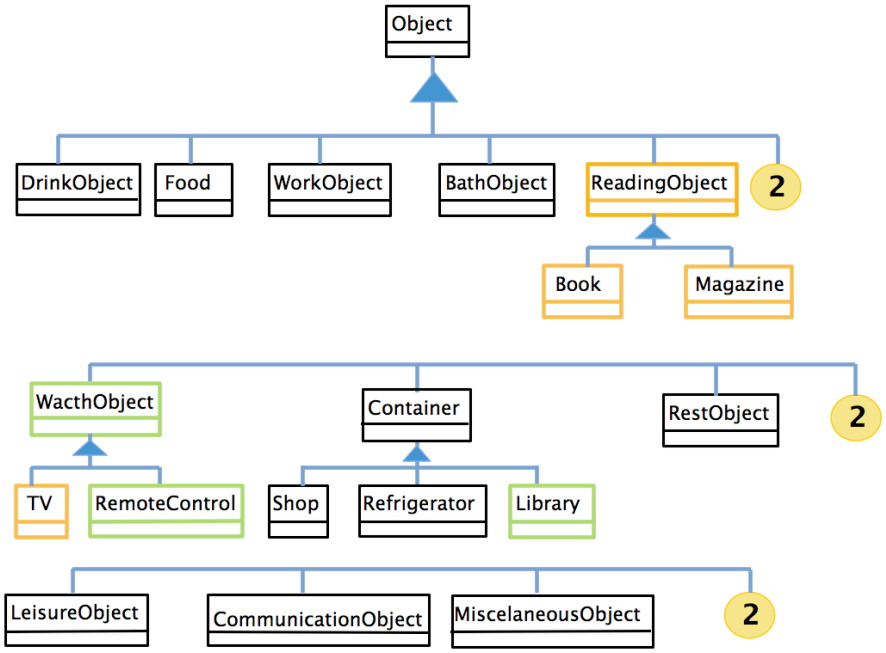
1) **Watching:** clase de tipo ClientAction creada para poder realizar la acción de ver la televisión que no esté vinculada al módulo de aprendizaje de habilidades.

2) **TellHability:** clase de tipo VerbalCommunication creada para poder realizar la acción de transmitir conocimiento entre dos actores. Esta acción está vinculada al módulo de aprendizaje de habilidades.

3) **ReadHability:** clase de tipo ClientAction creada para poder realizar la acción de leer un libro. Esta acción está vinculada al módulo de aprendizaje de habilidades.

4) **DoExercises;** clase de tipo ClientAction creada para poder realizar la acción de realizar ejercicios. Esta acción está vinculada al módulo de aprendizaje de habilidades.

5) **WatchingHability:** clase de tipo ClientAction creada para poder realizar la acción de ver la televisión. Esta acción está vinculada al módulo de aprendizaje de habilidades.



21. Clases que representan los objetos del escenario

Dentro de este archivo también se han creado nuevas clases para los nuevos objetos creados en el escenario. Las clases se encuentran representadas en el diagrama en color verde. Las clases que aparecen color naranja son las que se han visto modificadas, estas clases serán explicadas en el apartado ***4.3.2 Cambios realizados en modelo de conocimiento.*** A continuación se describen las clases creadas:

1) **Library:** clase de tipo Container que albergará los libros y las revistas del escenario.

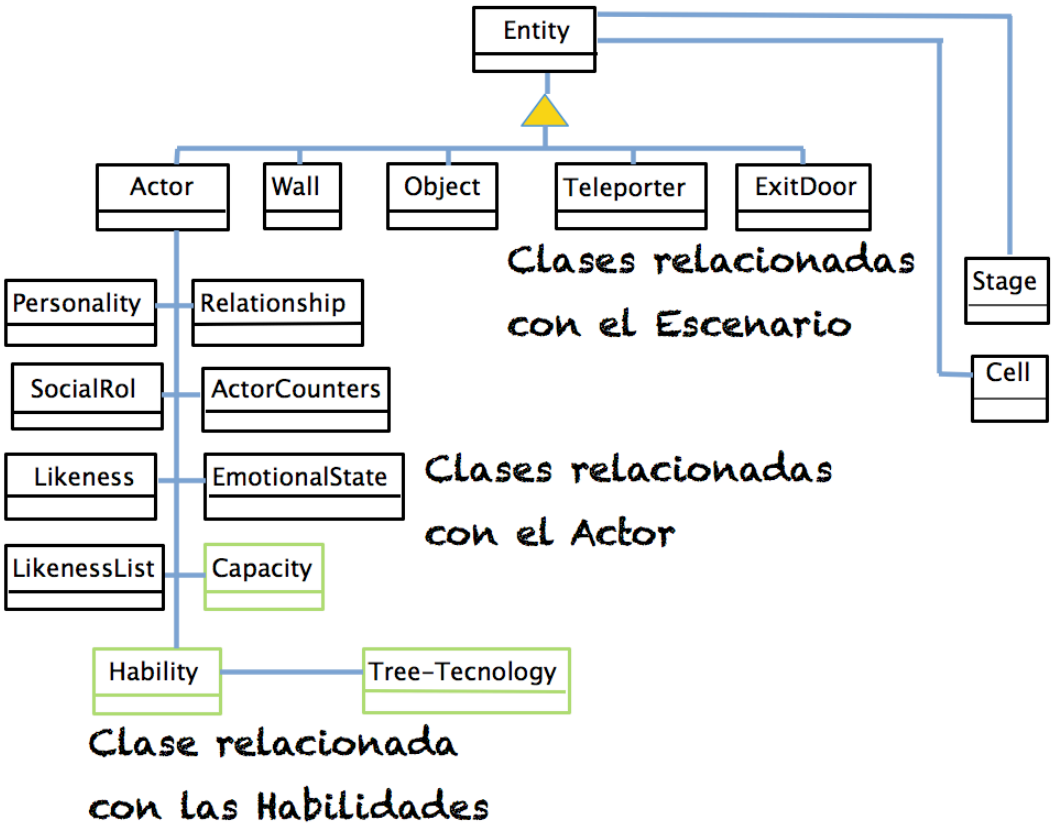
2) **WatchObject:** clase de tipo Object utilizada por los objetos de tipo visual. Esta clase ha sido creada para referirse a los diferentes objetos visuales, como por ejemplo la televisión o el mando de la televisión. Los campos que la forman son:

- **Type**: campo de tipo SYMBOL que sirve para categorizar los diferentes tipos de objetos visuales. En este caso sólo tenemos dos Tv y Mando.
- **Multislot accessCell**: campo de tipo INSTANCE-NAME utilizado para saber cual/es son las celda/as de acceso al objeto.

3) **RemoteControl:** clase de tipo WatchObject utilizada para la creación del objeto visual Mando. Dicho objeto será utilizado por las reglas de aprendizaje teórico en el que se vea implicado el objeto TV y Mando.

4.3.1.2 Aportación en "hability_ontology.clp"

Una vez realizado los cambios en el modelo de conocimiento inicial añadiendo todas las clases necesarias para la nueva funcionalidad. Para estas nuevas clases se decidió realizar el aporte en un nuevo fichero para facilitar su localización y mantenimiento de las mismas ya que no tienen dependencia con las que se encuentran dentro del anterior archivo. En el fichero también se encuentran las instancias que son necesarias para el buen funcionamiento del nuevo módulo de aprendizaje.



22. Diagrama de las clases principales añadidas a "hability_ontology.clp"

4.3.1.2.1 Clases creadas en "hability_ontology.clp"

Las nuevas clases creadas para poder integrar la nueva funcionalidad se han representado en verde en el anterior diagrama y se explican a continuación:

1) **Capacity**: clase de tipo USER utilizada para darle a un actor un tipo de capacidad concreta. Los campos que la forman son:

   - **actor:** campo de tipo INSTANCE-NAME que hace referencia al actor cuya capacidad se está creando. Este campo hace referencia a una instancia de tipo actor.

   - **Theorical**: campo de tipo INTEGER que puede tomar valores del rango 0 a 10. Este campo hace referencia a la capacidad de aprendizaje teórico que tiene un determinado actor.

   - **Practical**: campo de tipo INTEGER que puede tomar valores del rango 0 a 10. Este campo hace referencia a la

capacidad de aprendizaje práctico que tiene un determinado actor.

2) **Hability**: clase de tipo USER utilizada para la creación de las habilidades referentes a un actor. Esta clase será utilizada para crear las habilidades de un actor en concreto. Los campos que la forman son:

   - **actor**: campo de tipo INSTANCE-NAME que hace referencia al actor cuya habilidad se está creando. Este campo hace referencia a una instancia de tipo actor.
   - **Level**: campo de tipo INTEGER que puede tomar valores del rango 0 a 100. El campo indica el nivel en que la habilidad se encuentra.
   - **Type**: campo de tipo SYMBOL que representa el nombre de la habilidad.
   - **Available**: campo de tipo SYMBOL que indica si una habilidad se encuentra disponible para poder ser iniciada en el conocimiento.
   - **Complet**: campo de tipo SYMBOL que indica si la habilidad se encuentra completada. Una habilidad se encuentra completada en el momento que llega a su nivel máximo de aprendizaje.
   - **Achieved**: campo de tipo INTEGER que puede tomar valores del rango 0 a 2. Este campo hace referencia al número de habilidades conseguidas que se encuentran ligadas con esta habilidad. Se comparará con el campo "Necesary" para saber si la habilidad puede ser desbloqueada.
   - **Necesary**: campo de tipo INTEGER que puede tomar valores del rango 0 a 2. Este campo hace referencia al número de habilidades que necesita está habilidad para poder ser desbloqueada.
   - **Accumulated**: campo de tipo INTEGER que puede tomar valores del rango 0 a 10. Este campo indica el porcentaje de la habilidad aprendida. En el momento que llegue a

10, automáticamente aumentará en 1 el nivel de la habilidad.

- **Hability_required**: campo de tipo SYMBOL que indica la habilidad o habilidades que son necesarias para poder tener acceso a dicha habilidad. Este campo será utilizado para generar de forma automática aquellas habilidades que tienen ligado el árbol de tecnología.
- **Tree**: campo de tipo SYMBOL que indica si una habilidad tiene asociado o no un árbol de tecnología. Los valores que puede tener son:
  - **True**: indica que tiene un árbol de tecnología asociado.
  - **False**: indica que no tiene asociado ningún árbol de tecnología.
- **StateTree**: campo de tipo SYMBOL que indica el estado del árbol al que se encuentra ligada la habilidad. Existen tres posibles valores:
  - **INITIAL**: indica que el árbol aún no está listo para ser utilizado.
  - **PROGRESS**: indica que el árbol está siendo utilizado y que la habilidad aún no ha sido completada.
  - **COMPLETE**: indica que el árbol ya se ha completado.
  - **NEITHER**: indica que no tiene ningún árbol de tecnología asociado, por tanto no tiene estado.

3) **Program**: clase de tipo TV utilizada para la creación de canales de tv. Los campos que la forman son:

- **Watch**: campo de tipo SYMBOL que hace referencia al nombre del canal de tv.
- **Type**: campo de tipo SYMBOL que hace referencia al tipo de canal al que pertenece. Los valores que puede tomar son:

- **Hability**: valor que tomado por los programas que tienen asociado el aprendizaje de una habilidad.
- **Funny**: valor que tomado por los canales que sólo proporcionan diversión.

4) **Tree-Tecnology**: clase de tipo USER utilizada para la creación de un árbol de tecnología. Esta clase será utilizada para crear los diferentes tipos de niveles que componen el árbol. Un punto importante es saber que los árboles de tecnología son comunes para todos los actores. Los campos que la forman son:

   - **Id**: campo de tipo SYMBOL que identifica un Árbol de Tecnología.
   - **Hability**: campo de tipo SYMBOL que indica el nombre de la habilidad a la que hace referencia.
   - **Necesary:** campo de tipo INTEGER que indica el número de habilidades que necesita una habilidad para poder empezar a ser aprendida.
   - **Numec**: campo de tipo INTEGER que indica el número de habilidades, ligadas a una habilidad, que ya han llegado al nivel máximo. Esto es utilizado para no tener que llevar un orden concreto en el aprendizaje de las habilidades.
   - **Level**: campo de tipo INTEGER que indica el nivel en el que una habilidad se encuentra completada.

4.3.1.2.1 Instancias creadas en "hability_ontology.clp"

Las instancias creadas en este archivo son utilizadas por los actores para poder realizar actividades vinculas al aprendizaje. No todas las instancias utilizadas para esta nueva funcionalidad están creadas en este archivo, ya que solo se crean aquellas que no dependen directamente del actor.

1) **Program**: se creará una instancia por cada canal de televisión que exista.

2) **Tree-tecnology**: se creará una instancia por cada nivel del árbol de tecnología.

4.3.2 Cambios realizados en modelo de conocimiento

Los cambios realizados en el modelo de conocimiento se ha realizado sobre las clases ya existentes en el archivo "ontology.clp". Solamente se han creado nuevos cambios en las principales clases para poder adaptarlas a la nueva funcionalidad.

1) **AddClient**: clase de tipo ClientAction utilizada para agregar un actor al escenario. Esta clase ha sido modificada para poder adaptar las necesidades del aprendizaje a los actores, cuando estos van a ser introducidos dentro del escenario. Para ello hemos añadido los siguientes campos:

   - **Theoretical**: campo de tipo INTEGER que hace referencia a la capacidad de aprendizaje que tendrá el actor en el ámbito del aprendizaje Teórico. El valor que toma pertenece al rango de 0 a 10.

   - **Practical**: campo de tipo INTEGER que hace referencia a la capacidad de aprendizaje que tendrá el actor en el ámbito del aprendizaje Práctico. El valor que toma pertenece al rango de 0 a 10.

   - **Modo**: campo de tipo SYMBOL utilizado para poder determinar un modo de juego. Hay que destacar que en todos los tipos de modo de juego las acciones básicas como dormir, comer, beber y trabajar se realizan de forma completamente normal. Los valores que puede tomar son:

     - **Habilities**: campo multislot en el que aparecen las habilidades que no tienen asociadas un árbol de tecnología.

     - **Level_habilities**: campo multislot en el que aparecen los niveles que tendrán las habilidades que no tienen asociadas un árbol de tecnología que aparecen el campo Habilidad anteriormente descrito.

     - **Sociability**: valor que hace referencia al modo de aprendizaje de comunicación entre actores.

- **Hability**: valor utilizada valor que hace referencia al modo de aprendizaje de habilidades.
- **Rest**: valor utilizado para el modo normal.

2) **Container**: clase de tipo Object creada para los objetos que albergarán otro tipo de objetos. En esta clase se ha modificado el campo type para añadir un nuevo tipo de Container llamado librería. Dicho objeto será creado para albergar los libros y revistas del escenario. Dicha clase se ha explicado en el apartado de Modificaciones y creaciones en el fichero ONTOLOGY, en Clases creadas.

3) **ReadingObject**: clase de tipo Object utilizada por los objetos de tipo lectura. Esta clase ha sido modificada para poder cumplir con las necesidades de los nuevos objetos y poder así diferenciar los objetos de tipo lectura utilizados para el aprendizaje de las diferentes habilidades. Se han agregado los siguientes campos:

   - **Materials**: campo de tipo SYMBOL utilizado para saber a qué tipo de materia pertenece el objeto de lectura. Existen materias asociadas a habilidades y materias que no. Un ejemplo puede ser "Cocina" asociada a la habilidad Cocina y la materia "Magazine" que no tiene ninguna habilidad asociada.
   - **Exercises**: campo de tipo SYMBOL utilizado para saber si el objeto de tipo lectura contiene ejercicios o no.

4) **Tv**: clase de tipo WatchObject utilizada por la TV. Esta clase se ha visto modificada para poder adaptar las nuevas necesidades de los actores. En la versión anterior, esta clase no contaba con ningún campo, ahora está formada por:

   - **Program**: campo de tipo INSTANCE-NAME, utilizado para hacer referencia al canal que está siendo visto por un actor. El campo hace referencia a una instancia de tipo Program.
   - **On**: campo de tipo SYMBOL, utilizado para saber si la TV se encuentra encendida y por tanto disponible para ser vista desde cualquier punto de la habitación en la que se

encuentra, o si por lo contrario debería el actor de coger el mando y proceder a encenderla para poder ser usada.

5) **actor**: clase de tipo Entity utilizada para la creación de los actores. La clase se ha visto modificada por la necesidad de adecuar a los actores en el modulo de aprendizaje de habilidades. Los campos añadidos son:

   - **Capacity**: campo de tipo INSTANCE-NAME utilizado por la capacidad que tiene un actor para el aprendizaje. Este campo hace referencia a una instancia de tipo Capacity.

   - **Modo**: campo de tipo SYMBOL utilizado para poder determinar un modo de juego. Sus valores son, aunque hay que tener en cuenta que en todos los tipos de modo de juego las acciones básicas como dormir, comer, beber y trabajar se realizan de forma completamente normal:

     - **Hability**: valor que hace referencia al modo de aprendizaje de habilidades.
     - **Sociability**: valor que hace referencia al modo de aprendizaje de comunicación entre actores.
     - **Rest**: valor utilizado para el modo normal.

4.4 Desarrollo del servidor

El servidor está formado por diferentes archivos. Estos archivos han sido modificados para poder adaptar la nueva funcionalidad al juego AI-Live. A continuación se explicaran las aportaciones y los cambios que se han realizado en el servidor.

4.4.1 Cambios en el servidor

Las primeros cambios se han realizado en el servidor dentro del módulo de Inteligencia Artificial albergado en el archivo server.clp. En este módulo se encuentran las actividades comunes a todos los módulos, por ello he realizado cambios en las reglas generales para poder adaptarlas a la nueva funcionalidad, a continuación se explican los cambios:

1) **Read**: regla que permite leer un libro del escenario. La regla consta del siguiente antecedente:

   - Una instancia de la acción que vamos a realizar, en este caso de tipo Read.
   - Un escenario en el que debe encontrarse el actor.
   - Un actor que presente el modo habilidad activado. Dicho actor deberá tener cogido un objeto de tipo ReadingObject. Dicho objeto no se limitará sólo a los libros como en la regla anterior, sino que también están incluidos las revistas.
   - El conjunto de drivers asociados al actor (se encuentran en una instancia de la clase ActorCounters).
   - Una instancia de ReadingObject que contendrá todos los drivers que se verán modificados en el consecuente de la regla.

El consecuente de dicha regla es:

   - El incremento de los drivers Hunger y Tiredness.
   - Se eliminará la acción realizada.
   - Se modificará el turno de la acción para que de forma aleatoria se pueda realizar la acción hasta un máximo de tres veces seguidas.

2) **AddClient**: regla que permite añadir un nuevo actor al escenario. En esta regla se albergan todos los campos que son configurables para que cada actor pueda ser diferente. En este caso se ha visto modificado los siguientes campos:

   - Se han añadido los multisolt habilidad y nivel_habilidad para poder configurar aquellas habilidades que no tienen asignado ningún árbol de tecnología.

Las modificaciones en el consecuente son:

   - Se creará con ayuda de las funciones prong y f-index las instancias de las habilidades no ligadas a ningún árbol con su correspondiente nivel. La instancia de

habilidad presentará el valor NO para el campo tree y el valor NEITHER para el campo stateTree.

4.4.2 Aportaciones en el servidor

Para el correcto funcionamiento dl juego AI-Live se ha realizado diferentes aportaciones al servidor. El servidor consta de tres archivos, dos de los cuales ya existían. A continuación se explican las diferentes aportaciones al servidor para adaptar la nueva funcionalidad.

4.4.2.1 Aportaciones en server.clp

Las primeras aportaciones se han realizado en el servidor dentro del módulo de Inteligencia Artificial albergado en el archivo server.clp. En este módulo se encuentran las actividades comunes al resto de módulos. En este archivo solamente se ha realizado un aporte que es la regla Watching. Esta regla permite ver la TV a un actor, siempre y cuando la televisión se encuentre encendida y ambos estén en la misma habitación. La regla consta del siguiente antecedente:

- Una instancia de la acción que vamos a realizar, en este caso de tipo Watching.
- Un escenario en el que debe encontrarse el actor y la Televisión.
- Un actor que se encuentre en el mismo escenario que la tv.
- Una instancia de un Programa de televisión.
- Un objeto de tipo TV que debe de encontrarse encendida.
- El conjunto de drivers asociados al actor (se encuentran en una instancia de la clase ActorCounters).
- Una instancia de tipo WatchObject que contendrá todos los drivers que se verán modificados en el consecuente de la regla.

El consecuente de dicha regla es:

- El incremento de los drivers Dirtness, Boredom y Tiredness.

- Se eliminará la acción realizada.
- Se modificará el turno de la acción para que de forma aleatoria se pueda realizar la acción hasta un máximo de tres veces seguidas.

Se ha realizado dicho aporte ya que en el módulo de habilidades se realiza la misma actividad pero con el añadido del aprendizaje. Así se deja adaptado todo el servidor con la misma funcionalidad.

4.4.2.2 Aportaciones en hability_engine_server.clp

Para una mayor claridad se ha decidido crear un nuevo fichero servidor llamado "hability_engine_server.clp" para albergar todas las reglas del módulo de aprendizaje de habilidades. Esto ha sido así para poder localizar y manipular las reglas de dicho módulo de una forma más fácil y sencilla. A continuación se explicarán las reglas y funciones que lo forman:

4.4.2.2.1 Nuevas funciones

1) **capacity-hability-theorical**: función que permite realizar el cálculo del porcentaje acumulado de un actor por aprendizaje de una habilidad en modo Teórico. La regla consta del siguiente antecedente:
   - El porcentaje de la capacidad del actor para realizar el aprendizaje en modo teórico.
   - El porcentaje acumulado de la habilidad que se quiere aprender.
2) **capacity-hability-practical**: función que permite realizar el cálculo del porcentaje acumulado de un actor por aprendizaje de una habilidad en modo Práctico. La regla consta del siguiente antecedente:
   - El porcentaje de la capacidad del actor para realizar el aprendizaje en modo práctico.
   - El porcentaje acumulado de la habilidad que se quiere aprender.

4.4.2.2.1 Nuevas reglas

1) **ReadHability**: regla que permite leer un libro del escenario en modo habilidad. Esto significa que el actor al mismo tiempo que lee un libro aprende la habilidad asociada al mismo en modo teórico. La regla consta del siguiente antecedente:

   - Una instancia de la acción que vamos a realizar, en este caso de tipo ReadHability.
   - Un escenario en el que debe encontrarse el actor.
   - Un actor que presente el modo habilidad activado. Dicho actor deberá tener cogido un objeto de tipo ReadingObject, que deberá coincidir con alguna de sus habilidades activadas.
   - El conjunto de drivers asociados al actor (se encuentran en una instancia de la clase ActorCounters).
   - Una instancia de la capacidad del actor para que en el consecuente se pueda incrementar su conocimiento dependiendo de la capacidad del actor.
   - Una instancia de tipo ReadingObject que contiene los drivers que se van a ver modificados en el consecuente de la regla.

El consecuente de dicha regla es:

   - El incremento de los drivers Hunger y Tiredness.
   - El incremento del acumulado de la habilidad seleccionada en modo teórico.
   - Se eliminará la acción realizada.
   - Se modificará el turno de la acción para que de forma aleatoria se pueda realizar la acción hasta un máximo de tres veces seguidas.

2) **DoExercises**: regla que permite hacer ejercicios de un libro que se encuentre en el escenario en modo habilidad. Esto significa que el actor al mismo tiempo que hace ejercicios de

un libro aprende la habilidad asociada al mismo en modo práctico. La regla consta del siguiente antecedente:

- Una instancia de la acción que vamos a realizar, en este caso de tipo DoExercises.
- Un escenario en el que debe encontrarse el actor.
- Un actor que presente el modo habilidad activado. Dicho actor deberá tener cogido un objeto de tipo ReadingObject, que deberá coincidir con alguna de sus habilidades activadas. El objeto deberá tener la opción de hacer ejercicios, sino no podrá realizar dicha acción.
- El conjunto de drivers asociados al actor (se encuentran en una instancia de la clase ActorCounters).
- Una instancia de la capacidad del actor para que en el consecuente se pueda incrementar su conocimiento dependiendo de la capacidad del actor.
- Una instancia de tipo ReadingObject que contiene los drivers que se van a ver modificados en el consecuente de la regla.

El consecuente de dicha regla es:

- El incremento de los drivers Hunger y Tiredness.
- El incremento del acumulado de la habilidad seleccionada en modo práctico.
- Se eliminará la acción realizada.
- Se modificará el turno de la acción para que de forma aleatoria se pueda realizar la acción hasta un máximo de tres veces seguidas.

3) **WatchingHability**: regla que permite ver la televisión que se encuentre en el mismo escenario que un actor en modo habilidad. Esto significa que el actor al mismo tiempo que ve la televisión aprende la habilidad asociada al mismo en modo teórico. La regla consta del siguiente antecedente:

- Una instancia de la acción que vamos a realizar, en este caso de tipo WatchingHability.

- Un escenario en el que debe encontrarse el actor y la Televisión.
- Un actor que presente el modo habilidad activado. Dicho actor deberá tener cogido un objeto de tipo WatchingObject refiriéndose al mando.
- Una instancia de un Programa de televisión que deberá coincidir con alguna de sus habilidades activadas.
- El conjunto de drivers asociados al actor (se encuentran en una instancia de la clase ActorCounters).
- Una instancia de tipo WhatchingObject que contiene todos los drivers que se van a ver modificados en el consecuente de la regla.
- Una instancia de la capacidad del actor para que en el consecuente se pueda incrementar su conocimiento dependiendo de la capacidad del actor.

El consecuente de dicha regla es:

- El incremento de los drivers Dirtness, Boredom y Tiredness.
- Se eliminará la acción realizada.
- Se modificará el turno de la acción para que de forma aleatoria se pueda realizar la acción hasta un máximo de tres veces seguidas.
- El incremento del acumulado de la habilidad seleccionada en modo teórico.

4) **GenerateHabilities**: regla que permite crear instancias de la clase Hability de todas aquellas habilidades que tengan un árbol de tecnología asociado. La regla consta del siguiente antecedente:

   - Un escenario en el que se encuentre un actor.
   - El conjunto de drivers del actor.
   - Una instancia del árbol de tecnología de tipo Tree-Tecnology con una habilidad vinculada.

- Que no exista una instancia de la habilidad asociada al anterior árbol de tecnología.

El consecuente de dicha regla es:

- La creación de la instancia de la habilidad asociada al árbol de Tecnología. En dicha instancia la habilidad presentará el valor TRUE para el campo tree, el valor INITIAL para el campo stateTree, los valores por defecto para los campos level y accumulated, y las condiciones de dependencia que presente el árbol de tecnología como son las hability_requeried y las necesary propias de cada nivel.

4.4.2.2 Aportaciones en emotional_engine_server.clp

En este caso se ha decidido que la regla que permite que dos actores puedan realizar el intercambio de conocimientos mediante el diálogo se genere en él fichero donde se encuentran todas las capacidades de habla del actor. De esta manera se ha creado la regla TellHability dentro del fichero "emotional_engine_server". A continuación se realiza la explicación de la regla:

TellHability: Regla que permite el intercambio de conocimiento mediante el diálogo entre dos actores, siempre y cuando ambos actores se encuentre en la misma habitación. La regla consta del siguiente antecedente:

- Una instancia de actor que se encuentre en el escenario.
- Una instancia de otro actor que se encuentre en el mismo escenario que el anterior.
- Las instancias de los Drivers de los anteriores actores.
- Las instancias de Capacity de los anteriores actores.
- Las instancias de una Habilidad ligada a los dos anteriores actores, teniendo en cuenta siempre que uno de los actores deberá poseer un nivel superior de la habilidad elegida para el intercambio del conocimiento. La Habilidad elegida para el intercambio deberá encontrarse habilitada en ambos actores.

El consecuente de la regla es:

- Aumenta el acumulado de la **Habilidad** seleccionada para el intercambio de conocimiento en el **actor** que presenta el menor nivel.

4.5 Desarrollo de los clientes CLIPS

El juego AI-Live está compuesto por varios clientes como se han explicado en los anteriores apartados. Para la nueva funcionalidad solamente se han realizado cambios en los clientes CLIPS, tanto en el manual como en el de Inteligencia Artificial. Los cambios realizados son tanto modificaciones como aportaciones dentro de ambos archivos. Estos cambios se explicarán a lo largo de los siguientes apartados.

4.5.1 Cambios en los clientes CLIPS

Para poder adaptar los cliente CLIPS existentes a la nueva funcionalidad se han tenido que realizar ciertos cambios en las reglas que permiten que los actores realicen las acciones que deseen. Por ello he tenido que modificar algunas reglas que se explican a continuación.

1) **Read**: regla que permite leer un libro del escenario. La regla consta del siguiente antecedente:

   - Una instancia de la acción que vamos a realizar, en este caso de tipo Read.
   - Un escenario en el que debe encontrarse el actor.
   - Un ActorActor que presente el modo habilidad activado. Dicho actor deberá tener cogido un objeto de tipo ReadingObject. Dicho objeto no se limitará sólo a los libros como en la regla anterior, sino que también están incluidos las revistas.
   - El conjunto de drivers asociados al actor (se encuentran en una instancia de la clase ActorCounters).

   El consecuente de dicha regla será la activación de la acción de tipo Read, que activa la acción de leer junto al libro escogido. Se imprimirá la acción enviada al servidor y se imprimirán los drivers del actor en un fichero.

2) **ReadObjectGoal**: regla que permite la localización de un objeto de tipo "ReadObject" que se encuentra dentro de un Container, en este caso dentro de una librería. Su objetivo es moverse a la celda de acceso del objeto Container para poder cogerlo. La regla consta del siguiente antecedente:

   - Una instancia de la acción que vamos a realizar, en este caso MoveAction.
   - Una instancia de la clase Target que contiene el objetivo de la celda destino y un atributo para poder activar la instancia.
   - Una instancia del objeto Container, en este caso de una librería que contiene todos los libros y revistas del escenario.
   - El objeto de tipo ReadingObject refiriéndose tanto a libros como revistas.
   - La celda de acceso al objeto Container al que queremos ir. La celda de acceso será desbloqueada para que el actor pueda moverse hasta ella.

   El consecuente de dicha regla será la modificación de la instancia Target para que la regla del servidor se active y permita realizar el movimiento del actor hasta el objeto Container al que queremos ir, en este caso a la librería para poder coger un libro o revista.

4.5.2 Aportaciones en los clientes CLIPS

Para poder adaptar los cliente CLIPS existentes a la nueva funcionalidad se han tenido que realizar ciertos aportes para poder dar a los actores una serie de nuevas actividades que le puedan proporcionar el aprendizaje de las diferentes habilidades disponibles. Estos aportes se han realizado en forma de métodos y reglas.

4.5.2.1 Método creadas en el cliente

Los métodos creados son utilizados para mostrar por consola los cambios producidos por las diferentes reglas que proporcionan conocimiento.

1) **printWatchingClientAction()**: método permite imprimir en un fichero los datos referentes a la acción de tipo "Watching". El método es llamado por las reglas que permiten ver la televisión. Esta regla se llama "Watching" y será descrita en el siguiente apartado.

2) **printWatchingHabilityClientAction()**: método permite imprimir en un fichero los datos referentes a la acción de tipo WatchingHability. El método es llamado por la regla que permite ver la televisión y aprender al mismo tiempo. Esta regla se llama WatchingHability y será descrita en el siguiente apartado.

3) **printDoExercisesClientAction()**: método permite imprimir en un fichero los datos referentes a la acción de tipo DoExercises. El método es llamado por la regla que permite realizar al actor los ejercicios de un libro y así aprender de manera práctica una habilidad. Esta regla se llama DoExercises y será descrita en el siguiente apartado.

4) **printReadHabilityClientAction()**: método permite imprimir en un fichero los datos referentes a la acción de tipo ReadHability. El método es llamado por la regla que permite realizar la lectura de un libro y aprender de manera teórica una habilidad. Esta regla se llama ReadHability y será descrita en el siguiente apartado.

5) **printTellHabilityClientAction()**: método permite imprimir en un fichero los datos referentes a la acción de tipo TellHability. El método es llamado por la regla que permite transmitir conocimiento ente dos actores. Esta regla se llama TellHability y será descrita en el siguiente apartado.

4.5.2.2 Reglas creadas en el cliente

Los reglas creados son utilizados para realizar nuevas actividades que proporcionarán conocimiento a los actores una vez las realicen.

1) **TellHability**: regla que permite la interacción de dos actores para el intercambio de conocimiento. El actor que presente el nivel más alto de conocimiento transmitirá dicha habilidad al otro actor que aprenderá el conocimiento transmitido. La regla consta del siguiente antecedente:

    - Una instancia de un escenario en que se encontrarán los dos actores implicados en la regla.

- Una instancia de la acción que vamos a realizar, en este caso de tipo TellHability.
- Un actor que presente la habilidad que quiere transmitir.
- Un actor que recibe el conocimiento. Dicho actor tendrá que tener siempre un nivel inferior de conocimiento sobre la habilidad transmitida con respecto al otro actor. No es necesario que tenga un mínimo de conocimiento pero sí que dicha habilidad se encuentre disponible.
- Ambos personajes deben encontrarse en la misma habitación.
- El conjunto de drivers de ambos actores (se encuentran en una instancia de la clase ActorCounters).

El consecuente de dicha regla es la activación de los nuevos valores para el estado emocional de ambos actores y el listado de los drives de los actores en un fichero de trazas.

2) **ReadHability**: regla que permite leer un libro del escenario en modo habilidad. Esto significa que el actor al mismo tiempo que lee un libro aprende la habilidad asociada al mismo en modo teórico. La regla consta del siguiente antecedente:
   - Una instancia de la acción que vamos a realizar, en este caso de tipo ReadHability.
   - Un escenario en el que debe encontrarse el actor.
   - Un actor que presente el modo habilidad activado. Dicho actor deberá tener cogido un objeto de tipo ReadingObject, que deberá coincidir con alguna de sus habilidades activadas.
   - Una instancia de la habilidad seleccionada por el usuario para realizar el aprendizaje que debe de estar disponible y que debe de coincidir con el objeto que posee actualmente el usuario, en este caso un libro.

- El conjunto de drivers asociados al actor (se encuentran en una instancia de la clase ActorCounters).

El consecuente de dicha regla será la activación de la acción de tipo ReadHability, que activa la acción de leer aprendiendo junto al libro escogido. Se imprimirá la acción enviada al servidor y se imprimirán los drivers del actor en un fichero de trazas.

3) **DoExercises**: regla que permite hacer ejercicios de un libro que se encuentre en el escenario en modo habilidad. Esto significa que el actor al mismo tiempo que hace ejercicios de un libro aprende la habilidad asociada al mismo en modo práctico. La regla consta del siguiente antecedente:

   - Una instancia de la acción que vamos a realizar, en este caso de tipo DoExercises.
   - Un escenario en el que debe encontrarse el actor.
   - Un actor que presente el modo habilidad activado. Dicho actor deberá tener cogido un objeto de tipo ReadingObject, que deberá coincidir con alguna de sus habilidades activadas. El objeto deberá tener la opción de hacer ejercicios, sino podrá realizar dicha acción.
   - Una instancia de la habilidad seleccionada por el usuario para realizar el aprendizaje que debe de estar disponible y que debe de coincidir con el objeto que posee actualmente el usuario, en este caso un libro que contenga ejercicios para el aprendizaje práctico.
   - El conjunto de drivers asociados al actor (se encuentran en una instancia de la clase ActorCounters).

El consecuente de dicha regla será la activación de la acción de tipo DoExercises, que activa la acción de hacer ejercicios aprendiendo junto al libro escogido. Se imprimirá la acción enviada al servidor y se imprimirán los drivers del actor en un fichero de trazas.

4) **WatchingHability**: regla que permite ver la televisión que se encuentre en el mismo escenario que un actor en modo habilidad. Esto significa que el actor al mismo tiempo que ve la televisión aprende la habilidad asociada al mismo en modo teórico. La regla consta del siguiente antecedente:

   - Una instancia de la acción que vamos a realizar, en este caso de tipo WatchingHability.
   - Un escenario en el que debe encontrarse el actor y la Televisión.
   - Un actor que presente el modo habilidad activado. Dicho actor deberá tener cogido un objeto de tipo WatchingObject refiriéndose al mando.
   - Una instancia de un Programa de televisión que deberá coincidir con alguna de sus habilidades activadas.
   - Una instancia de la habilidad seleccionada por el usuario para realizar el aprendizaje que debe de estar disponible y que debe de coincidir con el objeto que posee actualmente el usuario, en este caso con el mando.
   - El conjunto de drivers asociados al actor (se encuentran en una instancia de la clase ActorCounters).

El consecuente de dicha regla será la activación de la acción de tipo WatchingHability, que activa la acción de ver la tv junto al programa seleccionado. Se imprimirá la acción enviada al servidor y se imprimirán los drivers del actor en un fichero de trazas.

5) **Watching**: regla que permite ver la televisión que se encuentre en el mismo escenario que un actor. Esto significa que el actor al mismo tiempo que ve la televisión aprende la habilidad asociada al mismo en modo teórico. La regla consta del siguiente antecedente:

   - Una instancia de la acción que vamos a realizar, en este caso de tipo Watching.

- Un escenario en el que debe encontrarse el actor y la Televisión.
- Un actor que se encuentre en el mismo escenario que la tv.
- Una instancia de un Program de televisión.
- El conjunto de drivers asociados al actor (se encuentran en una instancia de la clase ActorCounters).

El consecuente de dicha regla será la activación de la acción de tipo Watching que activa la acción de ver la tv junto al programa seleccionado. Se imprimirá la acción enviada al servidor y se imprimirán los drivers del actor en un fichero de trazas.

6) **WatchObjectGoal**: regla que permite la localización de un objeto de tipo WatchObject. Su objetivo es moverse a la celda de acceso del objeto visual, en este caso el objeto Mando, para que el actor pueda cogerlo posteriormente. La regla consta del siguiente antecedente:

   - Una instancia de la acción que vamos a realizar, en este caso MoveAction.
   - Una instancia de la clase Target que contiene el objetivo de la celda destino y un atributo para poder activar la instancia.
   - El objeto de tipo WatchObject refiriéndose solamente al objeto Mando.
   - La celda de acceso al objeto WatchObject al que queremos ir. La celda de acceso será desbloqueada para que el actor pueda moverse hasta ella.

El consecuente de dicha regla será la modificación de la instancia Target para que la regla del servidor se active y permita realizar el movimiento del actor hasta el objeto Container al que queremos ir, en este caso al lugar donde se encuentra el objeto Mando.

4.6 Desarrollo del escenario

Tras la creación del módulo de aprendizaje de habilidades, algunos objetos se han visto modificados con respecto a la versión anterior e incluso se han tenido que crear objetos nuevos. En los siguientes apartados se describen dichos cambios.

4.6.1 Cambios en el escenario

Para poder realizar las diferentes acciones vinculadas al aprendizaje, los actores deberán de utilizar ciertos elementos que se encuentran en los diferentes escenarios del juego AI-Live. Para ellos he tenido que realizar diferentes cambios.

1) **Books:** es un objeto de tipo **Book**. En la versión anterior se encontraba un solo objeto de este tipo que era utilizado para el disfrute del actor. Con la creación del nuevo módulo de aprendizaje se ha visto la necesidad de introducir en el escenario una serie de libros que satisfagan las necesidades del actor. Esto ha llevado a la eliminación del libro ya existente y se han introducido un libro por habilidad creada, hasta un total de siete libros.

   Todos los libros están contenidos dentro del objeto Librería que se encuentra situado dentro del escenario en las coordenadas (x,z,y 1,0,5), en vez de encontrarse solamente un libro en la mesilla de noche del actor.

   A continuación se nombran las diferentes instancias de los libros creados en el escenario:

   - **[Book_1_id]:** libro que hace referencia a la habilidad Cocina.
   - **[Book_2_id]:** libro que hace referencia a la habilidad Lógica.
   - **[Book_3_id]:** libro que hace referencia a la habilidad Física.
   - **[Book_4_id]:** libro que hace referencia a la habilidad Mecánica.

- **[Book_6_id]:** libro que hace referencia a la habilidad Informática.
- **[Book_7_id]:** libro que hace referencia a la habilidad Deporte.
- **[Book_8_id]:** libro que hace referencia a la habilidad Matemáticas.

2) **Magazine:** es un objeto de tipo Magazine. En la versión anterior se encontraba un solo objeto de este tipo que era utilizado para el disfrute del actor. En esta ocasión sólo se ha introducido un elemento más para ampliar su variedad, llegando a un total de dos revistas en el escenario.

   Todos las revistas están contenidas dentro del objeto Librería que se encuentra situado dentro del escenario en las coordenadas (x,z,y 1,0,5), en vez de encontrarse solamente una revista en la mesilla de noche del actor.

   A continuación se nombran las diferentes instancias de las revistas creadas en el escenario:

   - **[Book_5_id]:** revista de tipo Magazine.
   - **[Book_9_id]:** revista de tipo Newspaper.

4.6.2 Aportaciones en el escenario

El único aporte realizado en el escenarios es la creación de un nuevo tipo de contenedor llamado “Library”.

Library es un objeto de tipo Librería que a su vez es un objeto de tipo Container se ha creado para poder albergar los diferentes libros y revistas que se encuentran en el escenario. Se ha llegado a esta decisión porque se ha incrementado de forma razonable el número de objetos de tipo ReadingObject. Este objeto se encuentra situado en las siguientes coordenadas dentro del escenario (x,z,y 1,0,5).

5. Pruebas y resultados

Para poder comprobar que se ha producido la integración del nuevo módulo de aprendizaje, han sido necesarias dos tipos diferentes de pruebas: pruebas de funcionamiento y pruebas de integración.

Las pruebas de funcionamiento sirven para comprobar que las nuevas reglas implementadas funcionan correctamente. Estas pruebas han sido divididas en dos apartados que categorizan el tipo de reglas implementadas:pruebas para probar las reglas de actividades y pruebas para probar las reglas de habilidades.

Las pruebas de reglas de actividades consisten en ver el buen funcionamiento de las reglas que permiten al actor recoger y dejar los objetos necesarios para aprender, realizar las diferentes actividades que conllevan aprendizaje y las actividades de leer y ver la televisión como mera actividad de ocio. Para ello el actor utilizará objetos de aprendizaje tales como los objetos de lectura (libros y revistas) y objetos relacionados con la televisión como el mando a distancia. Las acciones de aprendizaje son leer aprendiendo, hacer ejercicios, ver la televisión aprendiendo y hablar con otro actor intercambiando conocimiento aprendido.

Para poder realizar estas actividades se realizarán comprobaciones en sus niveles y acumulados, observando que aumentan correctamente según se van realizando las diferentes actividades que conlleven un aprendizaje. Existirán actividades que durarán más de un turno, como es el caso del estudio de un libro o la realización de ejercicios. Esto es debido a que son actividades más largas que conllevan un aprendizaje y en un solo turno quedarían inconclusas. Para ello se utiliza la regla “CONTINUE”, que permite que un actor siga realizando la misma actividad durante uno o varios turnos, dichos turnos son establecidos con anterioridad en las condiciones del actor.

Estas pruebas también abarcan, aparte de actividades que conllevan un aprendizaje, reglas que permiten ver la televisión o leer un libro como actividad de ocio. La diferencia que existe entre leer un libro como actividad de ocio a leerlo como actividad de aprendizaje es el mero hecho de que un actor sólo podrá realizar su estudio si existe un libro que tenga una habilidad ligada y esta no se encuentre completada, en ese caso el actor podrá realizar la actividad, por lo contrario si no existe una habilidad o esta está completada solo podrá realizar la actividad de leerlo por diversión.

En el caso de ver la televisión las precondiciones son diferentes, un actor podrá ver la televisión como actividad de ocio siempre que la televisión se encuentre encendida y el actor esté en la misma habitación que la televisión. Para el caso de ver la televisión con fines de aprender, el actor deberá poseer el mando a distancia de la televisión y cambiar el canal de televisión a aquel canal que tenga ligada una habilidad que aun no tenga completada. Estas pequeñas diferencias hacen que un actor pueda proseguir con su aprendizaje realizando diferentes actividades e intercalándolas con otras actividades de ocio.

Las pruebas de reglas de habilidad consisten en comprobar que las reglas que se encargan de gestionar los niveles y acumulados de una habilidad, los diferentes niveles del árbol de tecnología, los estados de las habilidades o la creación automática de las habilidades ligadas a los diferentes niveles del árbol de tecnología salten cuando es debido.

Para poder realizar este tipo de pruebas el actor tendrá que ir pasando por las diferentes etapas en el aprendizaje de una habilidad. La primera regla que debe saltar es la creación automática de las habilidades ligadas a los diferentes árboles de tecnología, "GenerateHabilities", que generará por cada nuevo actor introducido en el juego en modo habilidad todas las habilidades necesarias para cubrir el árbol de tecnología. Una vez generada las diferentes habilidades, el actor podrá proceder al aprendizaje de las diferentes habilidades que se encuentren habilitadas, ya que dependiendo al nivel del árbol de tecnología al que pertenezcan éstas se encontrarán bloqueadas o habilitadas.

El actor irá completando el acumulado de una habilidad hasta llegar al máximo permitido, justo en este instante deberá saltar la regla "UpLevelHability" que permitirá que la habilidad aumente de nivel. Según vayan pasando los turnos y un actor consiga llevar el nivel de una habilidad a su máximo permitido es cuando deberá saltar la regla "CompleteHability", haciendo que una habilidad quede completada. Este sería el proceso convencional para dar por terminada una habilidad, pero no siempre es necesario que ocurra esto.

Cuando hablamos de habilidades ligadas o dependientes a un árbol de tecnología, nos referimos a habilidades que dependiendo al nivel que pertenezcan tendrán asociadas una o varias habilidades. En este caso para conseguir que se desbloquee un nivel de tecnología no es necesario que una habilidad llegué a su nivel máximo, sino que con que llegue a un nivel fijado en el árbol esté se desbloqueará. La regla que se encarga de realizar este proceso es "CompleteHabilityTree".

Pero no siempre es suficiente este paso, ya que en ciertos niveles existen más de una habilidad que bloquea al siguiente nivel del árbol de tecnología. Para estos casos se tiene en cuenta las habilidades dependientes haciendo que en el momento que una habilidad se da por completada salte la regla "AccumulateHability" que permite, una vez quede completada una habilidad y está a su vez sea necesaria para desbloquear una habilidad, haga que se elimine de su lista de necesarias. Esto hará que poco a poco el actor vaya desbloqueando el árbol de tecnología y consiga así llegar a completar su objetivo.

Por último una vez comprobado el buen funcionamiento de las reglas por separado se quiere comprobar que todas las reglas en conjunto interactúan adecuadamente, para ello se realizan las pruebas de integración. Estas pruebas consisten en recrear unos escenarios específicos y observar cómo se va desenvolviendo el actor para poder extraer y analizar un conjunto de datos. Una vez descritas las diferentes pruebas procedemos a desarrollarlas.

5.1. Pruebas de funcionamiento

A lo largo de la implementación del nuevo módulo ha sido necesario realizar una serie de pequeñas pruebas para poder comprobar el buen funcionamiento de las reglas.

Para ello he realizado:

a. **Comprobación de las reglas de actividades:** En este conjunto de pruebas se encuentran albergadas las reglas que hacen que un actor aprenda gracias a la interacción del actor con el medio, es decir,aquellas reglas que permiten a un actor realizar una actividad que le proporcione conocimiento. Las pruebas son las siguientes:

1º Prueba: *Comprobar que un actor recoge un objeto de lectura de la librería.*

Para poder realizar esta acción el actor hará uso de las reglas "ReadObjectGoal", "PickUpFromContainer" que le permiten detectar un objeto de tipo "ReadingObject" que se encuentra dentro de una librería y recogerlo. Una vez detectado el actor deberá proceder a recoger el objeto seleccionado de la librería. Cada objeto de tipo "ReadingObject" tiene un fin diferente, puede ser tanto el de enseñar como el de divertir.

En este caso no es indiferente que tipo de objeto es el elegido por el actor, ya que el objetivo es poder realizar la acción de recoger un objeto de tipo "ReadingObject".La traza realizada por esta prueba es la siguiente:

1) El actor localiza, en este caso, un periódico en la librería y decide cogerlo:

```
********** regla ReadObjectGoal **********

     - vamos a por: [Library_1_id]
     - y a por: [Book_9_id]
     - situado en x: 1 y: 5
     - coordenadas actuales del actor: X: 3 Y:15
     - celda de acceso al objeto : [DEFAULT_STAGE_0_2_5]

********** regla move **********

     - Se va a mover el actor con el Id: [CLIPS_ACTOR_LB8WBLZQOR]
     - Coordenadas futuras del actor: X= 2 Y= 5

     - Vamos a por objeto: [Library_1_id]

     - Vamos a por target: [Book_9_id]

********** regla PickUpFromContainer **********

     - objeto seleccionado : [Book_9_id]

     - target seleccionado : [Book_9_id]
```

2º Prueba: *Comprobar que el actor realiza una actividad de lectura tras coger un libro o revista del escenario. A continuación el actor deberá depositar el objeto dentro de la librería para que otro o el mismo actor pueda volver a utilizarlo.*

Para poder realizar esta acción deberá basarse en la anterior prueba, ya que es el paso previo, para poder continuar con la acción de dar uso al objeto de tipo "ReadingObject" cogido.

En este caso las reglas que interactúan con el actor son "Read" que permiten realizar una acción de lectura con un objeto de tipo "ReadingObject" que tenga un actor en su poder. Una vez seleccionado un objeto de tipo "ReadingObject", que en este caso será un libro ó una revista, el actor deberá llevar a cabo una acción de lectura.

Una vez realizada la acción el actor deberá dejar el objeto seleccionado en la librería de donde ha sido sustraído, para que él mismo u otro actor, pueda volver a darle uso. La traza realizada por esta prueba es la siguiente:

El actor, en este caso, localiza un periódico en la librería y decide cogerlo:

```
********** regla ReadObjectGoal **********

     - vamos a por: [Library_1_id]
     - y a por: [Book_9_id]
     - situado en x: 1 y: 5
     - coordenadas actuales del actor: X: 3 Y:15
     - celda de acceso al objeto : [DEFAULT_STAGE_0_2_5]

********** regla move **********

     - Se va a mover el actor con el Id: [CLIPS_ACTOR_LB8WBLZQOR]
     - Coordenadas futuras del actor: X= 2 Y= 5

     - Vamos a por objeto: [Library_1_id]

     - Vamos a por target: [Book_9_id]

********** regla PickUpFromContainer **********

     - objeto seleccionado : [Book_9_id]

     - target seleccionado : [Book_9_id]
```

1) El actor una vez recogido el periódico decide proceder a su lectura, y posteriormente devolverlo a la librería:

```
********** regla read **********

     - objecto de lectura  seleccionado : [Book_9_id]

     - El actor esta leyendo un libro de Newspaper

[Book_9_id] of Magazine
(stage [DEFAULT_STAGE])
(name_ "Book_9")
(x 6)
(y 6)
(z 0)
(sizeX 1)
(sizeY 1)
(sizeZ 1)
(angle north)
(weight 0)
(entityGraphic "cube")
(entityState "on")
(volume 0)
(accessCell [nil])
(value_hunger 1)
(value_thirst 0)
(value_tiredness 1)
(value_dirtness 0)
(value_boredom 0)
(value_solitude 0)
(type Magazine)
(materials Newspaper)
(exercises FALSE)
(in [CLIPS_ACTOR_LB8WBLZQOR])
********** regla PutDownInContainer **********

     - objeto dejado : [Book_9_id]

     - target seleccionado : [Book_9_id]
```

3º Prueba: *Comprobar que el actor realiza una actividad de lectura con aprendizaje tras coger un libro o revista del escenario. La actividad que puede realizar es la lectura o la realización de ejercicios. A continuación el actor deberá depositar el objeto dentro de la librería para que otro o el mismo actor pueda volver a utilizarlo.*

Para poder realizar esta acción el actor hará uso de las reglas “DoExercises” y “ReadHability” que le permiten dar un uso de aprendizaje al objeto de tipo “ReadingObject” escogido por el actor. Como en el caso anterior, esta prueba depende del buen funcionamiento de la primera, ya que es el paso previo y básico para poder realizar cualquier acción de este tipo.

A diferencia con la prueba anterior, el actor deberá darle un uso de aprendizaje al objeto escogido. Para ello podrá utilizar cualquiera de las reglas “DoExercises” y “ReadHability” que le permiten aprender de forma teórica o práctica.

Una vez realizada la acción el actor deberá dejar el objeto seleccionado en la librería de donde ha sido sustraído, para que él mismo u otro actor, pueda volver a darle uso. La traza realizada por esta prueba es la siguiente:

El actor localiza , en este caso, un libro de Lógica en la librería y procede a cogerlo:

```
********** regla ReadObjectGoal **********

   - vamos a por: [Library_1_id]
   - y a por: [Book_2_id]
   - situado en x: 1 y: 5
   - coordenadas actuales del actor: X: 6 Y:3
   - celda de acceso al objeto : [DEFAULT_STAGE_0_2_5]

********** regla move **********

   - Se va a mover el actor con el Id: [CLIPS_ACTOR_HBK71JISRQ]
   - Coordenadas futuras del actor: X= 2 Y= 5

   - Vamos a por objeto: [Library_1_id]

   - Vamos a por target: [Book_2_id]

********** regla PickUpFromContainer **********

   - objeto seleccionado : [Book_2_id]

   - target seleccionado : [Book_2_id]
```

1) El actor, en este caso, realiza un conocimiento en modo práctico con un libro de Lógica:

```
********** regla DoExercises **********

   - libro de ejercicios seleccionado : [Book_2_id]

   - Aumentado el accumulated de la habilidad Logic en modo practico

[Book_2_id] of Book
(stage [DEFAULT_STAGE])
(name_ "Book_2")
(x 6)
(y 6)
(z 0)
(sizeX 1)
(sizeY 1)
(sizeZ 1)
(angle north)
(weight 0)
(entityGraphic "cube")
(entityState "on")
(volume 0)
(accessCell [nil])
(value_hunger 1)
(value_thirst 0)
(value_tiredness 1)
(value_dirtness 0)
(value_boredom 0)
(value_solitude 0)
(type Book)
(materials Logic)
(exercises TRUE)
(in [CLIPS_ACTOR_HBK71JISRQ])
```

A continuación se mostrarían todas las habilidades que posee el actor, para evitar que se alarguen demasiado las trazas, he decidido mostrar solamente la habilidad que se va a modificar, en este caso la habilidad de Lógica:

```
********** Estas son las habilidades que tiene el cliente *********

([gen233] of Hability
(actor [CLIPS_ACTOR_HBK71JISRQ])
(level 0)
(type Logic)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 2)
(hability_required NEITHER)
(tree TRUE)
(stateTree COMPLETE)
)
```

En este caso la acción se ha realizado dos veces seguidas, de ahí que aparezca la regla CONTINUE, que permite volver a realizar la misma acción en el siguiente turno, este caso se puede dar hasta un máximo de 3 turnos:

```
********* regla CONTINUE **********

El Actor [CLIPS_ACTOR_HBK71JISRQ] continua con la accion anterior

********** regla PutDownInContainer **********

   - objeto dejado : [Book_2_id]

   - target seleccionado : [Book_2_id]
```

4º Prueba: *Comprobar que un actor recoge el objeto mando del escenario.*

Para poder realizar esta acción el actor hará uso de las reglas"WatchObjectGoal", "PickUpAction" y "PutDownAction" que le permiten detectar un objeto de tipo "WatchObject" y recogerlo del escenario. Una vez detectado el actor deberá proceder a recoger, en este caso el mando, del escenario. La traza realizada por esta prueba es la siguiente:

1) El actor selecciona el mando a distancia de la TV y decide ir a recogerlo del escenario:

```
********** regla  WatchingObjectGoal**********

   - vamos a por: [RemoteControl_1_id]
   - situado en x :6 y : 3
   - coordenadas actuales del actor: X: 2 Y: 5
   - celda de acceso al objeto : [DEFAULT_STAGE_0_6_3]

********** regla move **********

   - Se va a mover el actor con el Id: [CLIPS_ACTOR_HBK71JISRQ]
   - Coordenadas futuras del actor: X= 6 Y= 3

   - Vamos a por objeto: [RemoteControl_1_id]

   - Vamos a por target: [RemoteControl_1_id]
```

```
********** regla PickUpAction **********

    - objeto cogido : [RemoteControl_1_id]

    - catched : ()

    - target : [RemoteControl_1_id]

    - entities : ([CLIPS_ACTOR_HBK71JI5RQ] [Computer_1_id] [Sofa_1_id]
    [Refrigerator_1_id] [Bed_1_id] [Bed_2_id] [Phone_1_id] [Ball_1_id]
    [Library_1_id] [Ipod_1_id] [Guitar_1_id] [Shower_1_id] [Tv_1_id] [RemoteControl_1_id]
    [Bath_1_id] [Psp_1_id] [ExitDoor_1_id] [Teleporter_1_id] [DS_WALL_NORTH_ID]
    [DS_WALL_EAST_ID] [DS_WALL_SOUTH_ID] [DS_WALL_WEST_ID] [DS_FLOOR_ID])

objeto:
[RemoteControl_1_id] of RemoteControl
(stage [DEFAULT_STAGE])
(name_ "RemoteControl_1")
(x 6)
(y 3)
(z 0)
(sizeX 1)
(sizeY 1)
(sizeZ 1)
(angle south)
(weight 0)
(entityGraphic "RemoteControl")
(entityState "on")
(in [DEFAULT_STAGE_0_6_3])
(volume 0)
(value_hunger 0)
(value_thirst 0)
(value_tiredness 0)
(value_dirtness 0)
(value_boredom 0)
(value_solitude 0)
(value_wealth 50)
(type RemoteControl)
(accessCell)
```

5º Prueba: *Comprobar que un actor realiza la actividad de ver la televisión.*

En este caso la acción del actor hará uso de la regla "Watching" que le permiten dar un uso al objeto de tipo "WatchObject" que se encuentra dentro del escenario. Los usos que le puede dar pueden ser para realizar una acción que le proporcione unos conocimientos de aprendizaje o simplemente le proporcione diversión. Para este caso nos interesa probar que la acción de "ver la televisión" en modo diversión funciona correctamente. La traza realizada por esta prueba es la siguiente:

El actor decide ver la televisión en modo de diversión, por lo que lo único que necesita es estar en la misma habitación que la televisión y elegir un canal, en este caso ha decidido ver el canal comedia:

```
**********  regla WatchObjectGoal **********

     - Television  seleccionada : [Tv_1_id]

     - Programa  seleccionado Comedy_channel

[Tv_1_id] of Tv
(stage [DEFAULT_STAGE])
(name_ "tv_1")
(x 6)
(y 4)
(z 0)
(sizeX 1)
(sizeY 1)
(sizeZ 1)
(angle south)
(weight 2)
(entityGraphic "tv")
(entityState "on")
(in [nil])
(volume 0)
(value_hunger 0)
(value_thirst 0)
(value_tiredness 1)
(value_dirtness 12)
(value_boredom 1)
(value_solitude 0)
(type Tv)
(value_wealth 50)
(accessCell [DEFAULT_STAGE_0_6_5])
(On on)
```

6º Prueba: *Comprobar que un actor realiza la actividad de ver la televisión al mismo tiempo que aprende. A continuación el actor deberá depositar el objeto dentro del escenario para que otro o el mismo actor pueda volver a utilizarlo.*

Para poder realizar esta acción deberá basarse en la Prueba 4, ya que es el paso previo, para poder continuar con la acción de dar uso al objeto de tipo "WatchObject" cogido.

A diferencia con la prueba anterior, el actor deberá darle al objeto de tipo "WatchObject"seleccionado con anterioridad, un uso de aprendizaje. Para ello podrá utilizar la regla "WatchingHability" que le permiten aprender de forma teórica.

Una vez realizada la acción el actor deberá dejar el objeto seleccionado en el escenario del cuál ha sido sustraído, para que él mismo u otro actor, pueda volver a darle uso. La traza realizada por esta prueba es la siguiente:

El actor selecciona el mando a distancia de la TV y decide ir a recogerlo del escenario:

```
********** regla  WatchingObjectGoal**********

    - vamos a por: [RemoteControl_1_id]
    - situado en x :6 y : 3
    - coordenadas actuales del actor: X: 2 Y: 5
    - celda de acceso al objeto : [DEFAULT_STAGE_0_6_3]

********** regla move **********

    - Se va a mover el actor con el Id: [CLIPS_ACTOR_HBK71JISRQ]
    - Coordenadas futuras del actor: X= 6 Y= 3

    - Vamos a por objeto: [RemoteControl_1_id]

    - Vamos a por target: [RemoteControl_1_id]
```

Una vez adquirido el objeto, en este caso el Mando a distancia de la TV, procede a poner el canal y ver la televisión en modo teórico. En este caso el actor a elegido ver el Canal Cocina:

```
********** regla PickUpAction **********

    - objeto cogido : [RemoteControl_1_id]

    - catched : ()

    - target : [RemoteControl_1_id]

    - entities : ([CLIPS_ACTOR_HBK71JISRQ] [Computer_1_id] [Sofa_1_id]
    [Refrigerator_1_id] [Bed_1_id] [Bed_2_id] [Phone_1_id] [Ball_1_id]
    [Library_1_id] [Ipod_1_id] [Guitar_1_id] [Shower_1_id] [Tv_1_id] [RemoteControl_1_id]
    [Bath_1_id] [Psp_1_id] [ExitDoor_1_id] [Teleporter_1_id] [DS_WALL_NORTH_ID]
    [DS_WALL_EAST_ID] [DS_WALL_SOUTH_ID] [DS_WALL_WEST_ID] [DS_FLOOR_ID])

objeto:
[RemoteControl_1_id] of RemoteControl
(stage [DEFAULT_STAGE])
(name_ "RemoteControl_1")
(x 6)
(y 3)
(z 0)
(sizeX 1)
(sizeY 1)
(sizeZ 1)
(angle south)
(weight 0)
(entityGraphic "RemoteControl")
(entityState "on")
(in [DEFAULT_STAGE_0_6_3])
(volume 0)
(value_hunger 0)
(value_thirst 0)
(value_tiredness 0)
(value_dirtness 0)
(value_boredom 0)
(value_solitude 0)
(value_wealth 50)
(type RemoteControl)
(accessCell)
```

```
********** regla WatchObjectHability **********

   - Television  seleccionada : [Tv_1_id]

   - Programa  seleccionado Cooking en modo teorico

[Tv_1_id] of Tv
(stage [DEFAULT_STAGE])
(name_ "tv_1")
(x 6)
(y 4)
(z 0)
(sizeX 1)
(sizeY 1)
(sizeZ 1)
(angle south)
(weight 2)
(entityGraphic "tv")
(entityState "on")
(in [nil])
(volume 0)
(value_hunger 0)
(value_thirst 0)
(value_tiredness 1)
(value_dirtness 12)
(value_boredom 1)
(value_solitude 0)
(type Tv)
(value_wealth 50)
(accessCell [DEFAULT_STAGE_0_6_5])
(On on)
```

A continuación se mostrarían todas las habilidades que posee el actor, para evitar que se alarguen demasiado las trazas, he decidido mostrar solamente la habilidad que se va a modificar, en este caso la habilidad de Cocina:

```
********** Estas son las habilidades que tiene el cliente *********

([gen9] of Hability
(actor [CLIPS_ACTOR_HBK71JISRQ])
(level 2)
(type Cooking)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 1)
(hability_required Neither)
(tree FALSE)
(stateTree NEITHER)
)
```

Una vez terminada la acción el actor devolverá el mando a distancia de nuevo al escenario para poder ser utilizado en otro momento:

```
********** regla PutDownAction **********

   - objeto soltado : [RemoteControl_1_id]

   - catched : ([RemoteControl_1_id])

   - entities : ([CLIPS_ACTOR_HBK71JISRQ] [Computer_1_id] [Sofa_1_id]
    [Refrigerator_1_id] [Bed_1_id] [Bed_2_id] [Phone_1_id] [Ball_1_id] [Library_1_id]
    [Ipod_1_id] [Guitar_1_id] [Shower_1_id] [Tv_1_id] [Bath_1_id] [Psp_1_id] [ExitDoor_1_id]
    [Teleporter_1_id] [DS_WALL_NORTH_ID] [DS_WALL_EAST_ID] [DS_WALL_SOUTH_ID]
    [DS_WALL_WEST_ID] [DS_FLOOR_ID])

va a dejarlo en la celda [DEFAULT_STAGE_0_7_4]
coordenadas del actor x:6 y:3
info de la celda
[DEFAULT_STAGE_0_7_4] of Cell
(stage [DEFAULT_STAGE])
(occupied FALSE)
(inCell [nil])
(x 7)
(y 4)
(z 0)
```

7º Prueba: *Comprobar que un actor entabla una conversación con otro actor con la cuál realizan un intercambio de conocimientos.*

Para poder realizar esta acción el actor hará uso de las reglas "TellHability" que le permiten situarse en el mismo escenario que otro actor para poder establecer una conversación.

La conversación establecida entre los dos actores tendrá como finalidad el intercambio de conocimientos. El intercambio siempre se realizara del actor con más conocimiento al que menos conocimiento presente de la materia elegida para la conversación.

La traza realizada por esta prueba es la siguiente:

```
********* regla TELL Hability**********

   - El actor [CLIPS_ACTOR_XANSF738OT] le cuenta la habilidad Cooking [CLIPS_ACTOR_T178IUZVX2]
```

b. **Comprobación de las reglas de habilidad**: En este conjunto de pruebas se encuentran albergadas las reglas que hacen que el actor aprenda, es decir, aquellas reglas que saltan automáticamente una vez realizada una actividad y que no dependen directamente del actor. Las pruebas son las siguientes:

1º Prueba: *Comprobar que se han generado de forma automática todas las habilidades que tengan asociadas un árbol de tecnología.*

Para poder realizar esta prueba debe haber funcionado correctamente la regla "GenerateHabilities" que permite, una vez introducido un nuevo actor en el escenario, generar todas las habilidades que tiene asociadas el árbol de tecnología de un nuevo actor.

Esta comprobación se realiza viendo las trazas del programa a la hora de mostrar las habilidades que posee un actor. La traza realizada por esta prueba es la siguiente:

1) Se comprueban los diferentes niveles del árbol de tecnología:

```
([Tree_MATES] of TechnologyTree
  (id Tree_Maths)
  (hability Maths)
  (necesary Logic)
  (numnec 1)
  (level 1))

([Tree_Logic] of TechnologyTree
  (id Tree_Logic)
  (hability Logic)
  (necesary NEITHER)
  (numnec 0)
  (level 0))

([Tree_INFO] of TechnologyTree
  (id Tree_Informatics)
  (hability Informatics)
  (necesary Maths Physical)
  (numnec 2)
  (level 2))

([Tree_Physical] of TechnologyTree
  (id Tree_Physical)
  (hability Physical)
  (necesary Logic)
  (numnec 1)
  (level 1))
```

2) Una vez sabido que habilidades tienen que generase de forma automática, se busca en las trazas las habilidades generadas gracias a la regla "GenerateHabilities" y se comprueban que coincidan con las de los diferentes niveles del árbol de tecnología:

```
([gen231] of Hability
  (actor [CLIPS_ACTOR_HBK71JISRQ])
  (level 0)
  (type Informatics)
  (available FALSE)
  (complete FALSE)
  (achieved 0)
  (necesary 2)
  (accumulated 0)
  (hability_required Maths Physical)
  (tree TRUE)
  (stateTree INITIAL))

([gen232] of Hability
  (actor [CLIPS_ACTOR_HBK71JISRQ])
  (level 0)
  (type Maths)
  (available FALSE)
  (complete FALSE)
  (achieved 0)
  (necesary 1)
  (accumulated 0)
  (hability_required Logic)
  (tree TRUE)
  (stateTree INITIAL))
```

2º Prueba: *Comprobar que una vez aumentado el acumulado de un actor salte la regla que permite aumentar el nivel de la habilidad aprendida.*

Para poder realizar esta prueba debe haber funcionado correctamente la regla "UpLevelHability" que permite, una vez llegado al acumulado máximo de una habilidad, esté sea vaciado y por consiguiente aumentado en 1 el nivel de dicha habilidad aprendida.

Esta comprobación se realiza viendo las trazas del programa a la hora de mostrar las habilidades que posee un actor. La traza realizada por esta prueba es la siguiente:

1) El actor deberá tener una habilidad con un valor máximo del acumulado, para ello el actor ha realizado anteriormente diferentes acciones para conseguir dicho acumulado:

```
([gen233] of Hability
(actor [CLIPS_ACTOR_HBK71JISRQ])
(level 0)
(type Logic)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 10)
(hability_required NEITHER)
(tree TRUE)
(stateTree COMPLETE)
)
```

2) Una vez llegado al máximo deberá esperar un par de turnos para comprobar en las trazas que la regla "UpLevelHability" ha saltado y por tanto ha realizado bien su cometido:

```
([gen233] of Hability
(actor [CLIPS_ACTOR_HBK71JISRQ])
(level 1)
(type Logic)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required NEITHER)
(tree TRUE)
(stateTree COMPLETE)
)
```

***3º Prueba**: Comprobar que una vez llegado al máximo nivel de aprendizaje la habilidad queda "completada".*

Para poder realizar esta prueba debe haber funcionado correctamente la regla "CompleteHability" que permite, una vez llegado al máximo nivel de una habilidad, está se completa y no vuelve a saltar ninguna regla que lleve implicada esta habilidad en el modo de aprendizaje.

Esta comprobación se realiza viendo las trazas del programa a la hora de mostrar las habilidades que posee un actor. La traza realizada por esta prueba es la siguiente:

1) El actor deberá tener una habilidad en el nivel máximo, para ello el actor ha realizado anteriormente diferentes acciones para conseguir dicho nivel:

```
([gen10] of Hability
(actor [CLIPS_ACTOR_HBK71JISRQ])
(level 10)
(type Sport)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required Neither)
(tree FALSE)
(stateTree NEITHER)
)
```

2) Una vez llegado al máximo deberá esperar un par de turnos para comprobar en las trazas que la regla "CompleteHability" ha saltado y por tanto ha realizado bien su cometido:

```
([gen10] of Hability
(actor [CLIPS_ACTOR_HBK71JISRQ])
(level 10)
(type Sport)
(available TRUE)
(complete TRUE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required Neither)
(tree FALSE)
(stateTree NEITHER)
)
```

***4º Prueba**: Comprobar que una vez superado el nivel mínimo de conocimiento necesario para dar por superado un nivel del árbol de tecnología, éste sea superado.*

Para poder realizar esta prueba debe haber funcionado correctamente la regla "CompleteHabilityTree" que permite una vez llegado al conocimiento mínimo de un nivel del árbol de tecnología, dar por superado dicho nivel. Esto no significa que la habilidad queda completada, si no que nos dará paso a poder desbloquear parte de otros niveles del árbol de tecnología.

Esta comprobación se realiza viendo las trazas del programa a la hora de mostrar las habilidades que posee un actor. La traza realizada por esta prueba es la siguiente:

1) El actor deberá tener una habilidad en el nivel adecuado para quedar superado el nivel del árbol de tecnología, para ello el actor ha realizado anteriormente diferentes acciones para conseguir dicho nivel. En este caso el árbol seleccionado es el de Física:

```
([Tree_Physical] of TechnologyTree
  (id Tree_Physical)
  (hability Physical)
  (necesary Logic)
  (numnec 1)
  (level 1))
```

```
([gen234] of Hability
  (actor [CLIPS_ACTOR_HBK71JISRQ])
  (level 1)
  (type Physical)
  (available TRUE)
  (complete FALSE)
  (achieved 0)
  (necesary 0)
  (accumulated 0)
  (hability_required Logic)
  (tree TRUE)
  (stateTree PROGRESS)
)
```

2) Una vez llegado al nivel deseado se deberá esperar un par de turnos para comprobar en las trazas que la regla "CompleteHabilityTree" ha saltado y por tanto ha realizado bien su cometido:

```
([gen234] of Hability
  (actor [CLIPS_ACTOR_HBK71JISRQ])
  (level 1)
  (type Physical)
  (available TRUE)
  (complete FALSE)
  (achieved 0)
  (necesary 0)
  (accumulated 0)
  (hability_required Logic)
  (tree TRUE)
  (stateTree COMPLETE)
)
```

5º Prueba : *Comprobar que una vez se haya completado la/s habilidad/es requerida/s para un nivel del árbol de tecnología, dicho nivel del árbol se desbloquee.*

Para poder realizar esta prueba debe haber funcionado correctamente la regla "AccumulateHability" que permite, una vez quede completada una habilidad y está a su vez sea necesaria para desbloquear una habilidad, haga que se elimine de su lista de necesarias.

Esta comprobación se realiza viendo las trazas del programa a la hora de mostrar las habilidades que posee un actor. La traza realizada por esta prueba es la siguiente:

1) El actor deberá tener una habilidad completada, para ello el actor ha realizado anteriormente diferentes acciones para conseguir dicho nivel. En este caso el árbol seleccionado para desbloquear parte de sus necesarias es Informática, que necesita la habilidad

Matemáticas o Física para desbloquear parte del árbol:

```
([Tree_INFO] of TechnologyTree
  (id Tree_Informatics)
  (hability Informatics)
  (necesary Maths Physical)
  (numnec 2)
  (level 2))
```

```
([gen232] of Hability
  (actor [CLIPS_ACTOR_HBK71JISRQ])
  (level 10)
  (type Maths)
  (available TRUE)
  (complete TRUE)
  (achieved 0)
  (necesary 0)
  (accumulated 0)
  (hability_required Logic)
  (tree TRUE)
  (stateTree COMPLETE))
```

```
([gen231] of Hability
  (actor [CLIPS_ACTOR_HBK71JISRQ])
  (level 0)
  (type Informatics)
  (available FALSE)
  (complete FALSE)
  (achieved 0)
  (necesary 2)
  (accumulated 0)
  (hability_required Maths Physical)
  (tree TRUE)
  (stateTree INITIAL))
```

Una vez llegado nivel deseado, en este caso de la habilidad Matemáticas, deberá esperar un par de turnos para comprobar en las trazas que la regla "AccumulateHability" ha saltado y por tanto ha realizado bien su cometido. Como se pode ver se ha reducido en uno el número de habilidades necesarias y eliminado de su lista de habilidades requeridas la habilidad de Matemáticas:

```
([gen231] of Hability
  (actor [CLIPS_ACTOR_HBK71JISRQ])
  (level 0)
  (type Informatics)
  (available FALSE)
  (complete FALSE)
  (achieved 0)
  (necesary 1)
  (accumulated 0)
  (hability_required Physical)
  (tree TRUE)
  (stateTree INITIAL))
```

5.2 Pruebas de integración

Una vez comprobado el buen funcionamiento por separado de las reglas, el siguiente paso consiste en una serie de pruebas para ver si el conjunto de las reglas funcionan de forma adecuada.

Para ello, ha sido necesario crear una serie de escenarios concretos de los cuales poder extraer y analizar un conjunto de datos:

***1º Prueba**: se crea un escenario en el que cuenta con un actor con la configuración básica, es decir, cuenta con un perfil que no está vinculado al aprendizaje y conste con una capacidad normal para el desarrollo del aprendizaje.*

Esta prueba quiere ver como un actor con unos parámetros básicos o por defecto realiza la toma de decisiones a la hora de realizar actividades que conllevan aprender una habilidad, si realiza el aprendizaje sólo de una habilidad o las va intercalando con otras y qué modo de aprendizaje prefiere realizar.

El perfil utilizado es:

```
([DEFAULT_ACTOR] of AddClient
(name_ "Mike")
(gender M)
(entityGraphic "casual man")
(maxLoad 5)
(maxVolume 10)
(volume 1)
(weight 55)
(agreeableness 1.0)
(conscientousness 1.0)
(extraversion 1.0)
(neuroticism 0.0)
(openness 1.0)
(age 0)
(status 0)
(sex 0)
(arousal 0.0)
(valence 0.0)
(dislikenessList [Play] [Working])
(likenessList Bathtub Sofa Fruit Water [Resting])
(neutrallikeList Book)
(inter_likeList Meat [Washed] Wine)
(inter_dislikeList Bed Shower)
(maxhunger 50)
(maxthirst 70)
(maxdirtness 55)
(minwealth 100)
(maxtiredness 80)
(maxboredom 40)
(maxsolitude 75)
(continueProbability 95)
(theoretical 4)
(practical 2)
(modo Hability)
(habilities Cooking Sport)
(level_hability 2 5)
```

Como se puede observar el actor consta de gustos normales en los que no se destaca objetos relacionados con el aprendizaje y la capacidad de aprendizaje presenta valores intermedios.

Esto hará que el actor realice diferentes tareas y no sólo actividades relacionadas con el aprendizaje de habilidades.

Una vez lanzado el servidor y el actor se generan de forma correcta las habilidades ligadas al actor, tanto la de por defecto como las ligadas al árbol de tecnología.

Estos son los árboles:

```
([Tree_MATES] of TechnologyTree
  (id Tree_Maths)
  (hability Maths)
  (necesary Logic)
  (numnec 1)
  (level 1))

([Tree_Logic] of TechnologyTree
  (id Tree_Logic)
  (hability Logic)
  (necesary NEITHER)
  (numnec 0)
  (level 0))

([Tree_INFO] of TechnologyTree
  (id Tree_Informatics)
  (hability Informatics)
  (necesary Maths Physical)
  (numnec 2)
  (level 2))

([Tree_Physical] of TechnologyTree
  (id Tree_Physical)
  (hability Physical)
  (necesary Logic)
  (numnec 1)
  (level 1))
```

Estas son las habilidades:

```
([gen231] of Hability
  (actor [CLIPS_ACTOR_XANSF738OT])
  (level 0)
  (type Informatics)
  (available FALSE)
  (complete FALSE)
  (achieved 0)
  (necesary 2)
  (accumulated 0)
  (hability_required Maths Physical)
  (tree TRUE)
  (stateTree INITIAL))

([gen233] of Hability
  (actor [CLIPS_ACTOR_XANSF738OT])
  (level 1)
  (type Logic)
  (available TRUE)
  (complete FALSE)
  (achieved 0)
  (necesary 0)
  (accumulated 2)
  (hability_required NEITHER)
  (tree TRUE)
  (stateTree COMPLETE))

([gen232] of Hability
  (actor [CLIPS_ACTOR_XANSF738OT])
  (level 0)
  (type Maths)
  (available FALSE)
  (complete FALSE)
  (achieved 0)
  (necesary 1)
  (accumulated 0)
  (hability_required Logic)
  (tree TRUE)
  (stateTree INITIAL))

([gen234] of Hability
  (actor [CLIPS_ACTOR_XANSF738OT])
  (level 0)
  (type Physical)
  (available FALSE)
  (complete FALSE)
  (achieved 0)
  (necesary 1)
  (accumulated 0)
  (hability_required Logic)
  (tree TRUE)
  (stateTree INITIAL))

([gen9] of Hability
  (actor [CLIPS_ACTOR_XANSF738OT])
  (level 2)
  (type Cooking)
  (available TRUE)
  (complete FALSE)
  (achieved 0)
  (necesary 0)
  (accumulated 0)
  (hability_required Neither)
  (tree FALSE)
  (stateTree NEITHER))

([gen10] of Hability
  (actor [CLIPS_ACTOR_XANSF738OT])
  (level 5)
  (type Sport)
  (available TRUE)
  (complete FALSE)
  (achieved 0)
  (necesary 0)
  (accumulated 2)
  (hability_required Neither)
  (tree FALSE)
  (stateTree NEITHER))
```

Una vez comprobado esto, el actor empiece a desarrollar sus actividades. Según pasa el tiempo veo que tiene por prioridad el ver la televisión como modo de aprendizaje teórico. Estas son algunas de las trazas que muestran cómo va progresando en el aprendizaje de la habilidad Lógica, aunque también intercala esta habilidad con otras como Cocinar.

Usando la televisión como medio de aprendizaje:

```
********** regla WatchObjectHability **********

   - Television seleccionada : [Tv_1_id]

   - Programa seleccionado Logic en modo teorico

[Tv_1_id] of Tv
(stage [DEFAULT_STAGE])
(name_ "tv_1")
(x 6)
(y 4)
(z 0)
(sizeX 1)
(sizeY 1)
(sizeZ 1)
(angle south)
(weight 2)
(entityGraphic "tv")
(entityState "on")
(in [nil])
(volume 0)
(value_hunger 0)
(value_thirst 0)
(value_tiredness 1)
(value_dirtness 12)
(value_boredom 1)
(value_solitude 0)
(type Tv)
(value_wealth 50)
(accessCell [DEFAULT_STAGE_0_6_5])
(On on)
```

Usando los ejercicios de los libros como medio de aprendizaje práctico:

********** Estas son las habilidades que tiene el cliente *********
([gen233] of Hability
(actor [CLIPS_ACTOR_XANSF73BOT])
(level 0)
(type Logic)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 3)
(hability_required NEITHER)
(tree TRUE)
(stateTree COMPLETE)
)

********** regla DoExercises **********

- libro de ejercicios seleccionado : [Book_2_id]

- Aumentado el accumulated de la habilidad Logic en modo practico

[Book_2_id] of Book
(stage [DEFAULT_STAGE])
(name_ "Book_2")
(x 1)
(y 5)
(z 0)
(sizeX 1)
(sizeY 1)
(sizeZ 1)
(angle north)
(weight 0)
(entityGraphic "cube")
(entityState "on")
(volume 0)
(accessCell [nil])
(value_hunger 1)
(value_thirst 0)
(value_tiredness 1)
(value_dirtness 0)
(value_boredom 0)
(value_solitude 0)
(type Book)
(materials Logic)
(exercises TRUE)
(in [CLIPS_ACTOR_XANSF73BOT])

```
********** Estas son las habilidades que tiene el cliente *********

([gen233] of Hability
(actor [CLIPS_ACTOR_XANSF738OT])
(level 0)
(type Logic)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 9)
(hability_required NEITHER)
(tree TRUE)
(stateTree COMPLETE)
)
```

Según observo en las trazas el actor prefiere un desarrollo teórico a un desarrollo práctico. El tiempo estimado para terminar un nivel de habilidad ha sido superior a 40 minutos.

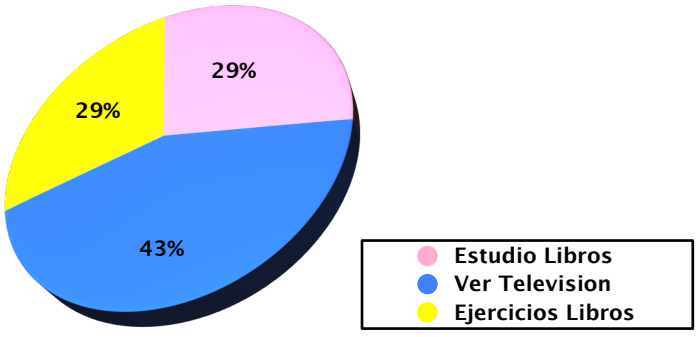
Ha desarrollado 7 actividades en total para completar el nivel de la habilidad Lógica. Las estadísticas de las actividades realizadas para conseguirlo son:

Modo teórico:

- Leyendo libros 2
- Viendo la televisión 3

Modo practico:

- Haciendo ejercicios en los libros 2



23. Resultado prueba de integración 1

- **Conclusión Prueba 1**: observando los resultados obtenidos de esta primera prueba puedo concretar que con la utilización de un perfil básico sin mostrar preferencias hacia el aprendizaje, el actor desarrollará la

habilidad, en este caso, Lógica en un periodo de tiempo de 43 minutos y utilizando 7 actividades diferentes. Esto significa que el objetivo del actor es el aprendizaje con la intercalación de otras actividades básicas y de ocio.

2º Prueba: *se crea un escenario en el que cuenta con un actor con una configuración orientada al aprendizaje, es decir sus gustos están vinculados con las actividades que proporcionan conocimiento, pero su capacidad para el conocimiento es básica.*

Esta prueba quiere ver como un actor con una serie de gustos orientados a las actividades que conllevan un aprendizaje de una habilidad y con una capacidad mínima de aprendizaje consigue completar un nivel de una habilidad.

El perfil utilizado es:

```
([DEFAULT_ACTOR] of AddClient
(name_ "Amy")
(gender F)
(entityGraphic "casual woman")
(maxLoad 5)
(maxVolume 10)
(volume 1)
(weight 55)
(agreeableness 1.0)
(conscientousness 1.0)
(extraversion 1.0)
(neuroticism 0.0)
(openness 1.0)
(age 0)
(status 0)
(sex 0)
(arousal 0.0)
(valence 0.0)
(dislikenessList  [Read] Book RemoteControl )
(likenessList Bathtub [Working] )
(neutrallikeList Ball)
(inter_likeList Sofa [Washed])
(inter_dislikeList Tv )
(maxhunger 10)
(maxthirst 10)
(maxdirtness 10)
(minwealth 30)
(maxtiredness 30)
(maxboredom 30)
(maxsolitude 30)
(continueProbability 40)
(theoretical 2)
(practical 2)
(modo Hability)
(habilities Cooking Sport)
(level_hability 2 5)
```

Como se puede observar el actor consta de gustos orientados a objetos utilizados a la hora de realizar un aprendizaje, tanto en modo teórico como práctico, la capacidad de aprendizaje que presenta tiene valores más orientados al aprendizaje práctico que teórico.

Esto hará que el actor prefiera realizar actividades básicas o de ocio frente a las actividades que conlleven aprender.

Una vez lanzado el servidor y el actor se generan de forma correcta las habilidades ligadas al actor, tanto la de por defecto como las ligadas al árbol de tecnología.

Estos son los árboles:

```
([Tree_MATES] of TechnologyTree
  (id Tree_Maths)
  (hability Maths)
  (necesary Logic)
  (numnec 1)
  (level 1))

([Tree_Logic] of TechnologyTree
  (id Tree_Logic)
  (hability Logic)
  (necesary NEITHER)
  (numnec 0)
  (level 0))

([Tree_INFO] of TechnologyTree
  (id Tree_Informatics)
  (hability Informatics)
  (necesary Maths Physical)
  (numnec 2)
  (level 2))

([Tree_Physical] of TechnologyTree
  (id Tree_Physical)
  (hability Physical)
  (necesary Logic)
  (numnec 1)
  (level 1))
```

Estas son las habilidades:

```
([gen247] of Hability
(actor [CLIPS_ACTOR_T178IUZVX2])
(level 2)
(type Cooking)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 6)
(hability_required Neither)
(tree FALSE)
(stateTree NEITHER)
)

([gen248] of Hability
(actor [CLIPS_ACTOR_T178IUZVX2])
(level 5)
(type Sport)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required Neither)
(tree FALSE)
(stateTree NEITHER)
)
```

```
([gen457] of Hability
(actor [CLIPS_ACTOR_T178IUZVX2])
(level 0)
(type Maths)
(available FALSE)
(complete FALSE)
(achieved 0)
(necesary 1)
(accumulated 0)
(hability_required Logic)
(tree TRUE)
(stateTree INITIAL)
)
([gen458] of Hability
(actor [CLIPS_ACTOR_T178IUZVX2])
(level 0)
(type Logic)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required NEITHER)
(tree TRUE)
(stateTree COMPLETE)
)
([gen459] of Hability
(actor [CLIPS_ACTOR_T178IUZVX2])
(level 0)
(type Physical)
(available FALSE)
(complete FALSE)
(achieved 0)
(necesary 1)
(accumulated 0)
(hability_required Logic)
(tree TRUE)
(stateTree INITIAL)
)
([gen460] of Hability
(actor [CLIPS_ACTOR_T178IUZVX2])
(level 0)
(type Informatics)
(available FALSE)
(complete FALSE)
(achieved 0)
(necesary 2)
(accumulated 0)
(hability_required Maths Physical)
(tree TRUE)
(stateTree INITIAL)
)
```

Una vez comprobado esto deja que el actor empiece a desarrollar sus actividades. Según pasa el tiempo veo que su comportamiento es similar al de la anterior prueba, con la diferencia que al presentar un nivel más bajo de capacidad de aprendizaje el actor tarda más en completar el nivel de la habilidad deseada. Estas son algunas de las trazas que muestran cómo va progresando en el aprendizaje de la habilidad Matemáticas, aunque también intercala esta habilidad con otras como Cocina o Lógica.

Usando los libros como medio de estudio:

```
********** regla DoExercises **********

    - libro de ejercicios seleccionado : [Book_8_id]

    - Aumentado el accumulated de la habilidad Maths en modo practico

[Book_8_id] of Book
(stage [DEFAULT_STAGE])
(name_ "Book_8")
(x 1)
(y 5)
(z 0)
(sizeX 1)
(sizeY 1)
(sizeZ 1)
(angle north)
(weight 0)
(entityGraphic "cube")
(entityState "on")
(volume 0)
(accessCell [nil])
(value_hunger 1)
(value_thirst 0)
(value_tiredness 1)
(value_dirtness 0)
(value_boredom 0)
(value_solitude 0)
(type Book)
(materials Maths)
(exercises TRUE)
(in [CLIPS_ACTOR_T178IUZVX2])

********** Estas son las habilidades que tiene el cliente *********
([gen457] of Hability
(actor [CLIPS_ACTOR_T178IUZVX2])
(level 0)
(type Maths)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 1)
(accumulated 8)
(hability_required NEITHER)
(tree TRUE)
(stateTree PROGRESS)
)
```

Usando los ejercicios de los libros como medio de aprendizaje práctico:

```
********** regla read-study **********

   - objecto de lectura  seleccionado : [Book_8_id]

   - Aumentado el accumulated de la habilidad Maths

[Book_8_id] of Book
(stage [DEFAULT_STAGE])
(name_ "Book_8")
(x 1)
(y 5)
(z 0)
(sizeX 1)
(sizeY 1)
(sizeZ 1)
(angle north)
(weight 0)
(entityGraphic "cube")
(entityState "on")
(volume 0)
(accessCell [nil])
(value_hunger 1)
(value_thirst 0)
(value_tiredness 1)
(value_dirtness 0)
(value_boredom 0)
(value_solitude 0)
(type Book)
(materials Maths)
(exercises TRUE)
(in [CLIPS_ACTOR_T178IUZVX2])



********** Estas son las habilidades que tiene el cliente *********
([gen457] of Hability
(actor [CLIPS_ACTOR_T178IUZVX2])
(level 0)
(type Maths)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 1)
(accumulated 3)
(hability_required NEITHER)
(tree TRUE)
(stateTree PROGRESS)
)
```

Según observo en las trazas el actor prefiere un desarrollo teórico a un desarrollo práctico. El tiempo estimado para terminar un nivel de habilidad ha sido superior a 60 minutos.

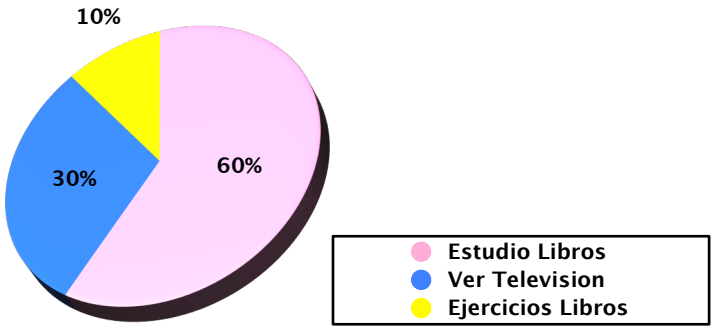
Ha desarrollado actividades en total para completar el nivel de la habilidad Lógica. Las estadísticas de las actividades realizadas para conseguirlo son:

Modo teórico:

- Leyendo libros 6
- Viendo la televisión 3

Modo practico:

- Haciendo ejercicios en los libros 1



24. Resultado prueba de integración 2

- **Conclusión Prueba 2**: observando los resultados obtenidos de en esta prueba puedo concretar que con la utilización de un perfil bajo sin mostrar ninguna preferencia ni gustos hacia el aprendizaje, el actor desarrollará la habilidad, en este caso, Matemáticas en un periodo de tiempo de 65 minutos y utilizando 10 actividades diferentes. Esto significa que el objetivo del actor es el aprendizaje con la intercalación de otras actividades básicas y de ocio que en este caso serán más abundantes que las de aprendizaje dado el número de actividades que realiza.

3º Prueba: *se crea un escenario que cuenta con un actor similar al anterior pero cuya capacidad para el conocimiento es alta.*

Esta prueba quiere ver como un actor con una serie de gustos orientados a las actividades que conllevan un aprendizaje de una habilidad y con una capacidad

alta de aprendizaje consigue completar un nivel de una habilidad.

El perfil utilizado es:

```
([DEFAULT_ACTOR] of AddClient
(name_ "Amy")
(gender F)
(entityGraphic "casual woman")
(maxLoad 5)
(maxVolume 10)
(volume 1)
(weight 55)
(agreeableness 1.0)
(conscientousness 1.0)
(extraversion 1.0)
(neuroticism 0.0)
(openness 1.0)
(age 0)
(status 0)
(sex 0)
(arousal 0.0)
(valence 0.0)
(dislikenessList  Bathtub [Working])
(likenessList [Read] Book RemoteControl)
(neutrallikeList Ball)
(inter_likeList Tv)
(inter_dislikeList Sofa [Washed])
(maxhunger 10)
(maxthirst 10)
(maxdirtness 10)
(minwealth 30)
(maxtiredness 30)
(maxboredom 30)
(maxsolitude 30)
(continueProbability 40)
(theoretical 6)
(practical 8)
(modo Hability)
(habilities Cooking Sport)
(level_hability 2 5)
```

Como se puede observar el actor consta de gustos orientados a objetos utilizados a la hora de realizar un aprendizaje, tanto en modo teórico como práctico,la capacidad de aprendizaje que presenta tiene valores más orientados al aprendizaje práctico que teórico.

Esto hará que el actor prefiera realizar actividades de aprendizaje frente a otras siempre que le sean posibles. Una vez lanzado el servidor y el actor se generan de forma correcta las habilidades ligadas al actor, tanto la de por defecto como las ligadas al árbol de tecnología.

Estos son los árboles:

```
([Tree_MATES] of TechnologyTree
  (id Tree_Maths)
  (hability Maths)
  (necesary Logic)
  (numnec 1)
  (level 1))

([Tree_Logic] of TechnologyTree
  (id Tree_Logic)
  (hability Logic)
  (necesary NEITHER)
  (numnec 0)
  (level 0))

([Tree_INFO] of TechnologyTree
  (id Tree_Informatics)
  (hability Informatics)
  (necesary Maths Physical)
  (numnec 2)
  (level 2))

([Tree_Physical] of TechnologyTree
  (id Tree_Physical)
  (hability Physical)
  (necesary Logic)
  (numnec 1)
  (level 1))
```

Estas son las habilidades:

```
([gen9] of Hability
(actor [CLIPS_ACTOR_XANSF738OT])
(level 2)
(type Cooking)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required Neither)
(tree FALSE)
(stateTree NEITHER)
)
([gen10] of Hability
(actor [CLIPS_ACTOR_XANSF738OT])
(level 5)
(type Sport)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 2)
(hability_required Neither)
(tree FALSE)
(stateTree NEITHER)
)
([gen231] of Hability
(actor [CLIPS_ACTOR_XANSF738OT])
(level 0)
(type Informatics)
(available FALSE)
(complete FALSE)
(achieved 0)
(necesary 2)
(accumulated 0)
(hability_required Maths Physical)
(tree TRUE)
(stateTree INITIAL)
)
([gen232] of Hability
(actor [CLIPS_ACTOR_XANSF738OT])
(level 0)
(type Maths)
(available FALSE)
(complete FALSE)
(achieved 0)
(necesary 1)
(accumulated 0)
(hability_required Logic)
(tree TRUE)
(stateTree INITIAL)
)
([gen233] of Hability
(actor [CLIPS_ACTOR_XANSF738OT])
(level 10)
(type Logic)
(available TRUE)
(complete TRUE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required NEITHER)
(tree TRUE)
(stateTree COMPLETE)
)
([gen234] of Hability
(actor [CLIPS_ACTOR_XANSF738OT])
(level 0)
(type Physical)
(availableTRUE)
(complete FALSE)
(achieved 0)
(necesary 1)
(accumulated 0)
(hability_required NEITHER)
(tree TRUE)
(stateTree PROGRESS)
)
```

Una vez comprobado esto, el actor empiece a desarrollar sus actividades. Según pasa el tiempo veo que tiene por prioridad hacer actividades relacionadas con los libros. Estas

son algunas de las trazas que muestran cómo va progresando en el aprendizaje de la habilidad “Física”, aunque también intercala esta habilidad con otras como Cocina o Sport.

Usando los ejercicios de los libros como medio de aprendizaje práctico:

```
********** regla DoExercises **********

   - libro de ejercicios seleccionado : [Book_3_id]

   - Aumentado el accumulated de la habilidad Logic en modo practic

[Book_2_id] of Book
(stage [DEFAULT_STAGE])
(name_ "Book_3")
(x 6)
(y 6)
(z 0)
(sizeX 1)
(sizeY 1)
(sizeZ 1)
(angle north)
(weight 0)
(entityGraphic "cube")
(entityState "on")
(volume 0)
(accessCell [nil])
(value_hunger 1)
(value_thirst 0)
(value_tiredness 1)
(value_dirtness 0)
(value_boredom 0)
(value_solitude 0)
(type Book)
(materials Physical)
(exercises TRUE)
(in [CLIPS_ACTOR_XANSF738OT])

********** Estas son las habilidades que tiene el cliente *********

([gen234] of Hability
(actor [CLIPS_ACTOR_XANSF738OT])
(level 0)
(type Physical)
(availableTRUE)
(complete FALSE)
(achieved 0)
(necesary 1)
(accumulated 8)
(hability_required NEITHER)
(tree TRUE)
(stateTree PROGRESS)
)
```

Según observo en las trazas el actor prefiere un desarrollo teórico a un desarrollo práctico. El tiempo estimado para terminar un nivel de habilidad ha sido aproximadamente 20 minutos.

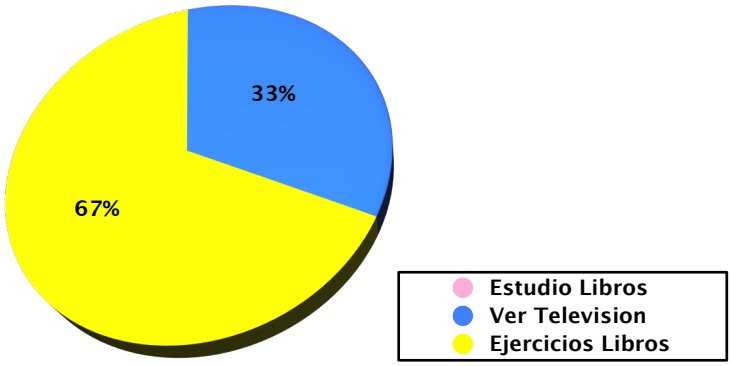
Ha desarrollado actividades 3 en total para completar el nivel de la habilidad. Las estadísticas de las actividades realizadas para conseguirlo son:

Modo teórico:

- Leyendo libros 0
- Viendo la televisión 1

Modo practico:

- Haciendo ejercicios en los libros 2



25. Resultado prueba de integración 3

- **Conclusión Prueba 3**: observando los resultados obtenidos de las anteriores pruebas puedo concretar que con la utilización de un perfil alto mostrando preferencia hacia el aprendizaje, el actor desarrollará la habilidad, en este caso, "Física" en un periodo de tiempo de 20 minutos y utilizando 3 actividades. Esto significa que el actor al presentar una mayor capacidad de aprendizaje realiza un número inferior de actividades y por tanto el tiempo empleado para conseguir su objetivo es menor que en los dos anteriores casos.

***4º Prueba**: se crea un escenario que cuenta con dos actores para comprobar si interfieren entre ellos las diferentes actividades de aprendizaje y si se produce un intercambio de conocimiento entre ellos.*

Esta prueba quiere ver como dos actor interactúan en un mismo escenario tras haber conseguido realizar un aprendizaje básico por separado. Con ello se busca la interacción de dos actores que transmiten el conocimiento aprendido a otro actor que carece de él.

Para poder realizar la prueba el juego debe estar ejecutando dos actores distintos, en este caso estos son los dos perfiles utilizados para realizar la prueba:

```
([DEFAULT_ACTOR] of AddClient
(name_ "Amy")
(gender F)
(entityGraphic "casual woman")
(maxLoad 5)
(maxVolume 10)
(volume 1)
(weight 55)
(agreeableness 1.0)
(conscientousness 1.0)
(extraversion 1.0)
(neuroticism 0.0)
(openness 1.0)
(age 0)
(status 0)
(sex 0)
(arousal 0.0)
(valence 0.0)
(dislikenessList  Bathtub [Working])
(likenessList [Read])
(neutrallikeList Ball)
(inter_likeList Tv)
(inter_dislikeList Sofa [Washed])
(maxhunger 10)
(maxthirst 10)
(maxdirtness 10)
(minwealth 30)
(maxtiredness 30)
(maxboredom 30)
(maxsolitude 30)
(continueProbability 40)
(theoretical 4)
(practical 2)
(modo Hability)
(habilities Cooking Sport)
(level_hability 2 5)
```

```
([DEFAULT_ACTOR] of AddClient
(name_ "Mike")
(gender M)
(entityGraphic "casual man")
(maxLoad 5)
(maxVolume 10)
(volume 1)
(weight 55)
(agreeableness 1.0)
(conscientousness 1.0)
(extraversion 1.0)
(neuroticism 0.0)
(openness 1.0)
(age 0)
(status 0)
(sex 0)
(arousal 0.0)
(valence 0.0)
(dislikenessList [Play] [Working])
(likenessList Bathtub Sofa Fruit Water [Resting])
(neutrallikeList Book)
(inter_likeList Meat [Washed] Wine)
(inter_dislikeList Bed Shower)
(maxhunger 50)
(maxthirst 70)
(maxdirtness 55)
(minwealth 100)
(maxtiredness 80)
(maxboredom 40)
(maxsolitude 75)
(continueProbability 95)
(theoretical 4)
(practical 2)
(modo Hability)
(habilities Cooking Sport)
(level_hability 6 9)
```

A continuación se corrobora que se han generado correctamente los árboles de tecnología y las habilidades asociadas a cada actor.

1) Los árboles de tecnología son los mismos para ambos actores por lo que solo se generan una vez:

([Tree_MATES] of TechnologyTree
(id Tree_Maths)
(hability Maths)
(necesary Logic)
(numnec 1)
(level 1))

([Tree_Logic] of TechnologyTree
(id Tree_Logic)
(hability Logic)
(necesary NEITHER)
(numnec 0)
(level 0))

([Tree_INFO] of TechnologyTree
(id Tree_Informatics)
(hability Informatics)
(necesary Maths Physical)
(numnec 2)
(level 2))

([Tree_Physical] of TechnologyTree
(id Tree_Physical)
(hability Physical)
(necesary Logic)
(numnec 1)
(level 1))

2) Las habilidades que se han generado son:

a) actor “Amy” [CLIPS_Actor_RIMSUDEWNB] :

([gen9] of Hability
(actor [CLIPS_ACTOR_GYSLLDGHW0])
(level 2)
(type Cooking)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required Neither)
(tree FALSE)
(stateTree NEITHER)
)
([gen10] of Hability
(actor [CLIPS_ACTOR_GYSLLDGHW0])
(level 5)
(type Sport)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required Neither)
(tree FALSE)
(stateTree NEITHER)
)
([gen231] of Hability
(actor [CLIPS_ACTOR_GYSLLDGHW0])
(level 0)
(type Informatics)
(available FALSE)
(complete FALSE)
(achieved 0)
(necesary 2)
(accumulated 0)
(hability_required Maths Physical)
(tree TRUE)
(stateTree INITIAL)
)

([gen232] of Hability
(actor [CLIPS_ACTOR_GYSLLDGHW0])
(level 0)
(type Maths)
(available FALSE)
(complete FALSE)
(achieved 0)
(necesary 1)
(accumulated 0)
(hability_required Logic)
(tree TRUE)
(stateTree INITIAL)
)
([gen233] of Hability
(actor [CLIPS_ACTOR_GYSLLDGHW0])
(level 0)
(type Logic)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required NEITHER)
(tree TRUE)
(stateTree COMPLETE)
)
([gen234] of Hability
(actor [CLIPS_ACTOR_GYSLLDGHW0])
(level 0)
(type Physical)
(available FALSE)
(complete FALSE)
(achieved 0)
(necesary 1)
(accumulated 0)
(hability_required Logic)
(tree TRUE)
(stateTree INITIAL)
)

b) actor “Mike” [CLIPS_Actor_GYSLLDGHW0]:

```
([gen457] of Hability
(actor [CLIPS_ACTOR_RIMSUDEWNB])
(level 0)
(type Logic)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required NEITHER)
(tree TRUE)
(stateTree COMPLETE)
)

([gen460] of Hability
(actor [CLIPS_ACTOR_RIMSUDEWNB])
(level 0)
(type Physical)
(available FALSE)
(complete FALSE)
(achieved 0)
(necesary 1)
(accumulated 0)
(hability_required Logic)
(tree TRUE)
(stateTree INITIAL)
)

([gen247] of Hability
(actor [CLIPS_ACTOR_RIMSUDEWNB])
(level 6)
(type Cooking)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required Neither)
(tree FALSE)
(stateTree NEITHER)
)

([gen458] of Hability
(actor [CLIPS_ACTOR_RIMSUDEWNB])
(level 0)
(type Informatics)
(available FALSE)
(complete FALSE)
(achieved 0)
(necesary 2)
(accumulated 0)
(hability_required Maths Physical)
(tree TRUE)
(stateTree INITIAL)
)

([gen248] of Hability
(actor [CLIPS_ACTOR_RIMSUDEWNB])
(level 9)
(type Sport)
(available TRUE)
(complete FALSE)
(achieved 0)
(necesary 0)
(accumulated 0)
(hability_required Neither)
(tree FALSE)
(stateTree NEITHER)
)

([gen459] of Hability
(actor [CLIPS_ACTOR_RIMSUDEWNB])
(level 0)
(type Maths)
(available FALSE)
(complete FALSE)
(achieved 0)
(necesary 1)
(accumulated 0)
(hability_required Logic)
(tree TRUE)
(stateTree INITIAL)
)
```

Una vez comprobado que todo se ha generado correctamente, se comprueba que se ha producido el intercambio de una habilidad entre dos actores.

En este caso la habilidad intercambiada se realiza al instante de introducir el segundo actor en el escenario, ya que como se puede observar en perfil mostrado con anterioridad, la habilidad Deporte y la habilidad Cocina muestran diferentes niveles para facilitar la ejecución de esta regla. En este caso la habilidad intercambiada es Deporte:

```
********* regla TELL Hability**********

- El actor [CLIPS_ACTOR_GYSLLDGHW0]le cuenta la habilidad Sport a [CLIPS_ACTOR_RIMSUDEWNB]
```

Según observo en las trazas la ejecución de la regla se realiza al siguiente turno tras introducirse el segundo actor en el escenario. Esto es debido a que ya existía de antemano una habilidad común a ambos actores que presentaban niveles diferentes.

Al comprobar este comportamiento decidí realizar la misma prueba, pero está vez poniendo los mismos niveles por defecto para las habilidades de ambos actores. Para este caso el comportamiento fue similar, en el momento que uno de los actores completaba el nivel de habilidad, al siguiente turno ejecutaba la regla "TellHability".

- **Conclusión Prueba 4**: observando los resultados obtenidos en las dos pruebas realizadas se puede observar que en el instante en el que un actor presenta una habilidad con nivel superior con respecto al otro actor, se ejecuta la regla "TellHability" al siguiente turno. Esto demuestra que un actor tiene preferencia en realizar un intercambio de conocimiento con un actor, a realizar un aumento de la habilidad utilizando objetos del escenario.

6. Presupuesto

Este apartado tiene como finalidad documentar una primera estimación de los costes de la realización del proyecto.

Para ello se ha estudiado los costes de proyectos anteriores para tener una medida más aproximada de los costes que se producen en un proyecto de esta envergadura.

6.1 Etapas del proyecto

Para el desarrollo de este proyecto he tenido que dividirlo en diferentes fases. Estas fases conllevan un tiempo determinado en el desarrollo del proyecto que conllevan unos gastos.

Fases	Horas
Análisis	120
Diseño	100
Implementación	287
Pruebas	170
Documentación	200

1. Tabla etapas del proyecto

6.2 Costes de personal

Para poder realizar este proyecto he tenido que adoptar diferentes roles a la hora de enfrentarme a las diferentes fases anteriormente descritas.

A continuación se presentan los costes de los diferentes roles adaptados para el proyecto, todas las unidades se presentan en Euros (€):

Rol	Analista	Ingeniero	Programador
Coste mes	3.040	2.720	1.989
Salario neto/mes	1.824	1.632	1.492
Salario/Hora	11,4	10,2	9,325
Coste Hora	19	17	15

2. Tabla Costes de personal

A continuación se explica el cálculo del coste para un empleado de la empresa como resumen para la tabla anterior. Tomaremos como ejemplo el Analista (se va a calcular el salario mensual y el coste total

que la empresa asume por el empleado, pero más adelante se tendrán en cuenta sólo las horas que realmente se van a emplear en el proyecto, no todo el salario):

El Analista tiene un sueldo mensual de neto 1.824€ al mes, o 11,4€ netos/hora, teniendo en cuenta que un mes tiene 160 horas laborables (40 horas semanales).

Por lo tanto y para conocer la retención que hay que aplicar a dicho sueldo obtenemos el sueldo anual: 1.824€/mes * 14 pagas (meses) = 25.536€/año.

Aplicando la tabla de tramos del IRPF observamos que hay que aplicar una retención del 37% lo que suman un total de 34.048€ brutos anuales. Si repartimos ese sueldo bruto en las 14 pagas obtenemos un salario bruto mensual de 2.432€/mes.

A dicho sueldo bruto la empresa tiene que asumir los costes asociados a la seguridad social y a las dietas lo que hacen un 20% sobre el coste bruto de cada persona física, en el ejemplo del Analista la empresa asume unos costes mensuales de 3.040€/mes, o 19€/hora.

Una vez calculado el coste por hora que le supone a la empresa cada empleado, tomaremos sólo las horas que realmente cada miembro va a trabajar en el presente proyecto, que va a ser 1 hora diaria (lo que es lo mismo, 4 horas por semana, pues se trabajará solo de lunes a jueves), lo que hace un total de 16 horas mensuales.

6.3 Costes de material

Se asumen en los costes de material aquellos que están asociados a material de uso tales como los ordenadores que se necesitarán para llevar a cabo el proyecto, por lo tanto:

Se asignan 2 ordenadores de 1.240€/Ordenador los cuales tienen un plazo de amortización de 3 años debido a esto se tiene en cuenta un coste de 34,4€/mes y por ordenador lo que hacen un total de 2 Ordenadores * 6 Meses estimados de proyecto * 42€ /mes hacen un total de 413,3€ en concepto de material.

Además se asigna una impresora para la impresión de los documentos, ésta tiene un coste de 800€ con un plazo de amortización de 3 años se obtiene un coste por mes de 11,2€/mes, dado que tenemos una estimación de 6 meses se tiene un coste de la impresora de 11,2€/mes * 6 meses hacen un total de 67,2€ en concepto de material.

Se obtiene un total de 480,53€ en concepto de materiales tangibles:

	Empresa / mes	Coste total
Coste	€ 80,1	€ 480,53

3. Tabla Concepto de materiales tangibles

6.4 Costes varios (indirectos)

A continuación se detallan los costes relacionados con el material fungible (CDs, folios, etc.) y los gastos relacionados con el agua y la luz. Se ha tenido en cuenta que se producen unos gastos de aproximadamente un 25% del coste total lo que asciende a un total de 480,53€ * 0,25 = 120,2€.

6.5 Costes totales

Una vez calculados los costes de los diferentes recursos se va a realizar la suma total de los mismos para hacer la estimación de lo que ha costado realizar el proyecto.

A continuación se presentan los costes desglosados en los diferentes apartados , todas las unidades se presentan en Euros (€):

Recursos	Coste
Personal	4,948
Material	480,53
Indirectos	120,2
Total	5548,73

4. Tabla Costes totales

Cabe destacar que debido a la normativa de la Universidad Carlos III de Madrid no se han contabilizado el margen de riesgos y beneficios para este proyecto, teniendo en cuenta solamente los gastos generados para el desarrollo del mismo.

7. Conclusiones del proyecto realizado

AI-Live es un proyecto conjunto realizado por diferentes alumnos de la universidad Carlos III de Madrid. Mi aportación al proyecto ha sido la creación de un nuevo módulo que permite a los actores desarrollar diferentes habilidades a partir de las capacidades que presenta cada actor.

La realización de este proyecto me ha aportado una nueva visión sobre el concepto de los videojuegos al poder indagar en los comienzos y ver todo lo que se ha logrado con el aporte de las nuevas tecnologías.

Partiendo de unos orígenes muy humildes la cultura de los videojuegos evoluciono a unos pasos gigantescos gracias a la intuición y creatividad de sus grandes pioneros. Fueron hombres que entendieron lo mucho que a la humanidad le gusta y necesita jugar, abrieron las puertas a un mundo maravilloso que una vez que se entra no se puede salir. Ver como utilizando simples algoritmos como los que utilizo William Higinbothan en 1958 para amenizar las visitas de de la División de Instrumentación del Laboratorio Nacional Brookheaven se ha podido pasar a crear la tecnología “Kinect” creada por Alex Kipman para Microsoft que permite el reconocimiento de objetos para interactuar y poder jugar con ellos. Pensar que sólo se trataba de un mero entretenimiento para algunos cuando para otros es el poder avanzar e investigar conceptos nuevos aportando ideas que se utilizan para el día a día de la sociedad.

La inteligencia artificial es una pequeña base en este mundo que cada día va cogiendo más y más fuerza ganando la aceptación en la sociedad en la que hoy vivimos. Gracias a la investigación y a la evolución en las nuevas tecnologías podemos disfrutar de videojuegos que plasman el comportamiento de la sociedad en la que hoy vivimos.

El desarrollo de este proyecto tenia en sí la adaptación de uno de los comportamientos más naturales del ser humano, el aprendizaje. He intentado adaptar este comportamiento al sistema AI-Live basándolo en el aprendizaje de habilidades comunes que reflejan todos las persona.

Los objetivos de mi proyecto se han visto cumplidos ya que he conseguido crear e integrar un nuevo módulo dentro del proyecto AI-Live. Los objetivos que debía de cumplir mi proyecto eran:

1) La creación de un módulo en el que los actores pudieran desarrollar una serie de habilidades a partir de las capacidades que presenta cada actor, haciendo que el aprendizaje no sea el mismo para todos si no que dependa de cada actor en concreto.

2) La creación de un árbol de tecnología que permitiese crear dependencias entre las diferentes habilidades y así hacer que los actores desarrollasen un objetivo.

3) Integrar los diferentes módulos existentes en AI-Live a mi nuevo módulo de habilidades.

Tras la realización de las pruebas, he podido observar que el aprendizaje está ligado a las capacidades y a la predisposición que presenta el actor a los objetos que se encuentran en el escenario. Esto proporciona una mayor personalización, haciendo que los actores aunque sean tratados por igual puedan desarrollarse de forma independiente y personalizada.

El nuevo módulo ha proporcionado a los actores una mayor similitud con el comportamiento humano, ampliando las actividades que pueden realizar, haciendo así que no sólo se trate de realizar actividades básicas o secuencias lógicas, sino que pueda realizar actividades con una finalidad, en este caso el aprendizaje.

Cómo último matiz, cabe destacar que durante la realización del proyecto hemos coincidido más de un alumno en la implementación de código haciendo más complicado el desarrollo. Gracias a la utilización de un sistema de control de versiones que ofrece Google de manera gratuita, “Google Code”, se ha podido trabajar de una forma ordenada, teniendo controlados los cambios realizados por cada alumno.

Una vez finalizado el proyecto, el sistema AI-Live ofrece una mayor similitud con los juegos de simulación de vida social como “Los Sims”, dejando abierta la puerta para que se puedan crear diferentes lineas de trabajo. En el siguiente apartado describo algunas de las líneas que se pueden desarrollar para futuros proyectos de final de carrera.

8. Líneas futuras de trabajo

AI-Live es un proyecto conjunto que ha sido creado por varios alumnos de la universidad Carlos III de Madrid como proyecto de final de carrera. Al tratarse de un proyecto conjunto cada alumno ha aportado una nueva visión al juego ofreciendo diferentes líneas de trabajo. Una vez finalizado mi proyecto existen varias líneas de trabajo que puede seguir AI-Live en futuras aportaciones.

8.1 Desarrollo de carácter y personalidad

En la actualidad AI-Live presenta la posibilidad de que los actores tengan algunos parámetros de personalidad, pero no que dichos parámetros puedan influenciar de forma directa en las diferentes actividades que realiza un actor.

Una posible vía de desarrollo sería que dependiendo de la personalidad que presente un actor, éste realice ciertas actividades mejor que otras, o que con ciertos objetos del escenario puedan perfeccionar diferentes aspectos de personalidad, haciendo por ejemplo que un actor tímido pueda practicar delante de un espejo para mejorar la comunicación con otros actores.

La personalidad o el carácter de una persona es un aspecto importante en la vida real, por lo que los actores en el juego AI-Live deberían de serlo también. Si se realizase esta línea de trabajo, los actores podrían ser mas diferentes los unos de los otros, haciendo que el juego se asemeje más al comportamiento real.

8.2 Carreras profesionales

En la actualidad AI-Live sólo cuenta con la posibilidad de realizar un trabajo para poder ganar dinero y así poder comprar alimentos para poder sobrevivir. Con la aportación del módulo de habilidades se podría relacionar las habilidades con diferentes trabajos, pudiendo desarrollar una vida profesional.

Esta idea haría que según se vayan realizando las habilidades y subiendo de nivel se pudieran desarrollar nuevos trabajos proporcionando diferentes ingresos y desbloqueando nuevas habilidades. Si se realizase esta línea de trabajo les podría dar a los actores más posibilidades laborales y no sólo trabajar con el ordenador

desde casa, pudiendo incluso realizar otras actividades en otros escenarios.

8.3 Mejora de la actividades

En la actualidad AI-Live dispone de una serie de actividades que puede desarrollar un actor para proporcionarle cansancio, felicidad, dinero, saciedad… Estas actividades estaban ligadas a que un actor tuviese cierto objeto en su poder, por ejemplo para leer el actor debía de tener en su poder un libro o una revista.

Con la incorporación del nuevo módulo de habilidades se puede desarrollar otra línea de trabajo en la que sea necesaria ciertos conocimientos previos para poder realizar ciertas actividades. Por ejemplo un actor que posea un nivel 1 de mecánica podría desatascar un desagüe pero no podría arreglar un grifo, ya que éste debería poseer el nivel 2, y tampoco podría arreglar una bicicleta ya que para ello debería poseer un nivel 5 de mecánica y un nivel 2 de físico.

Si se generase este tipo de necesidades para desarrollar ciertas actividades se logrará que los actores puedan ampliar sus conocimientos realizando diferentes actividades prácticas que le puedan ayudar a solucionar problemas que le surjan en la vida cotidiana asemejando aun más el juego a la vida real.

8.4 Metas personales

Una vez terminado mi proyecto AI-Live dispone de un árbol de tecnología que tiene diferentes niveles en los que se crean las dependencias para poder desarrollar las diferentes habilidades. Este concepto se puede utilizar para que un actor tenga ciertas metas personales que deba cumplir para alcanzar sus objetivos.

Las metas pueden ser desde realizar ciertas actividades como formar una familia o llegar al nivel máximo en su trabajo. Las metas proporcionarían a los actores un objetivo a cumplir, con una serie de necesidades que deberían de ir cumpliendo a lo largo del juego. Esto proporcionaría al actor cumplir sus metas y no solo realizar actividades con el fin de sobrevivir.

8.5 Relaciones personales con otros actores

En la actualidad AI-Live dispone de multijugador pudiendo realizar diferentes actividades entre los actores que se encuentren en el mismo escenario. La relación existente entre los actores es nula, solamente se relacionan para transmitir conocimiento, hablar sobre gustos o cotillear pero sin existir una vinculación directa entre ellos.

La nueva línea de trabajo podría estar vinculada a que los diferentes actores que se encuentren en el escenario puedan presentar una relación entre ellos, tanto sea predeterminada, como conseguida con la realización de ciertas actividades. Si se realizase esta línea de trabajo se podrían crear lazos entre los actores llegando a realizar actividades conjuntas para mejorar su relación, haciendo que incluso puedan crearse familias o relaciones personales.

8.5 Actualizar el apartado gráfico

En la actualidad AI-Live dispone de un cliente gráfico (GUI) que proporciona la interfaz al juego. Este cliente se encuentra desactualizado con la incorporación de nuevos objetos haciendo que ciertas actividades no se puedan ver correctamente.

En el cliente gráfico se podrían realizar ciertas mejoras aparte de actualizarlo con los nuevos objetos. La nueva línea de trabajo podría mejorar la visualización con las cámaras en los escenarios pequeños ya que la visualización no es muy acertada. También se podría mejorar la forma de representar los drivers de los actores, ya que cuando se está utilizando la opción de multijugador los drivers de los diferentes actores no se ven con claridad ya que se salen del recuadro asignado para ellos. Este problema se podría resolver haciendo que el jugador pudiera seleccionar un actor y mostrar en cada instante sólo los drivers del actor seleccionado y no todos.

8.6 Interacción con otros escenarios

En la actualidad AI-Live dispone de dos escenarios diferenciados, la tienda y la casa del actor. Esto hace que sólo se puedan realizar actividades dentro de la vivienda y en el caso de la tienda se pueda ir a comprar ciertos objetos de alimentación.

Esta línea de trabajo plantea la creación de diferentes escenarios para poder incrementar el número de actividades a realizar. Se podría

crear espacios abiertos como jardines o parques en la que los actores puedan realizar actividades conjuntas con otros actores, como puede ser la jardinería o la práctica de diferentes deportes como la natación o el tenis. Se pueden recrear diferentes tipos de tiendas en la que los actores puedan comprar ropa, accesorios o muebles para sus casas.

8.7 Realización de actividades a tiempo real

En la actualidad AI-Live dispone de un sistema de turnos en la los tiempos de los actores se gestionan en 1 o más turnos, dependiendo de la actividad que realicen. Esta línea de trabajo iría vinculada a cambiar la arquitectura del sistema para poder gestionar las actividades en tiempo real y no en turnos como se encuentra en la actualidad, consiguiendo así que sea un sistema multiagente. El cambio sería bastante laborioso pero a la larga beneficiaría al juego en el desarrollo de las actividades, ya que abriría nuevas líneas de trabajo, como que existieran los días, años... haciendo que los actores pudieran envejecer y asemejarse aun más al comportamiento humano.

9. Bibliografía

Documentación de Proyectos de Fin de Carrera previos:

- **[P.REZ, 2006]** Proyecto de Fin de Carrera: AI-Live. Miguel Alfonso Pérez Bonomini. Universidad Carlos III de Madrid. 2006.

- **[BENITO, 2007]** Proyecto de Fin de Carrera: Necesidades básicas de actores en universos de realidad simulada. Miguel Benito García. Universidad Carlos III de Madrid. 2007.

- **[JIM.NEZ, 2008]** Proyecto de Fin de Carrera: Dise.o e implementación de un modelo comunicativo emocional para agentes virtuales en el universo AI-Live. Marta Jiménez Matarranz. Universidad Carlos III de Madrid. 2008.

- **[ESCUDERO, 2011]** Proyecto de Fin de Carrera: Ampliación y mejora del universo virtual AI-Live. Javier Escudero Moreno. Universidad Carlos III de Madrid. 2011.

- **[UCEDA, 2012]** Proyecto de Fin de Carrera:Desarrollo e implementación de un cliente para el entorno virtual de AI-Live. Alberto Uceda Blanco. Universidad Carlos III de Madrid. 2012.

Bibliografía ordenada alfabéticamente por el apellido del primer autor:

• D. Borrajo, N. Juristo, V. Martínez y J. Pazos, "Inteligencia Artificial. Métodos y Técnicas", Centro de Estudios Universitarios Ramón Areces, Madrid, 1993.

• Castle, L. 1998. "The Making of Blade Runner, Soup to Nuts!" In Proceedings of the Computer Game Developers' Conference, Long Beach, CA, 87-97.

• Castronova, E. "Synthetic Worlds: The Business and Culture of Online Games", University of Chicago Press, 2005.

• Laird, J. E. 2000. It Knows What You're Going To Do: Adding Anticipation to a Quakebot. In Papers from the AAAI 2000 Spring Symposium on Artificial Intelligence and Interactive Entertainment, Technical Report SS-00-02, 41-50. AAAI Press.

Webs ordenadas por el nombre de la web:

- Capital SIMS. Historia de Maxis, 1987-2007. Visitado en Septiembre de 2012: http://sims.capitalsim.net/articulo/Historia_de_Maxis,_1987_-_2007.

- Espadas y dados. Fable: The Lost Chapters. Visitado en Agosto de 2012: http://espadasydados.blogspot.com.es/2011/06/fable-lost-chapters.html.

- Gran angular. Inteligencia Artificial. Departamento de Lenguajes y Sistemas Informáticos UPC-FIB. IA Algoritmos de Juegos. Visitado en Septiembre de 2012: http://www.gran-angular.net/wp-content/uploads/2008/07/algoritmos-de-juegos.pdf.

- Histpry of Computers, hadware, software, internet... Leonardo Torres's chess-machine. Visitado en Junio de 2012: http://history-computer.com/Dreamers/Torres_chess.html.

- Icons of Progress. Deep Blue. Visitado en Agosto de 2012: http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/.

- IDELab. Universidad de Valladolid. Algoritmo A*. Visitado en Noviembre de 2012: http://idelab.uva.es/algoritmo.

- Innova. Universidad Nacional de Educación a Distancia. Inteligencia Artificial: Introducción y Tareas de Búsqueda. Versión 20100619. Robert J. de la Fuente López. Visitado en Junio de 2012: http://www.innova.uned.es/webpages/aconute/iartificial/documentos/ia_intro_busqueda.pdf.

- Información Departamento de Informática. Universidad de Valladolid. Apuntes: MINIMAX y poda Alfabeta. . Visitado en Julio de 2012: http://www.infor.uva.es/~arancha/IA/busqueda/MINIMAX%20y%20poda%20alfabeta.pdf.

- Macuarium. Black&White. Visitado en Agosto de 2012: http://www.macuarium.com/macuarium/actual/especiales/2002_06_17_bw.shtml.

- PDP-1 Restoration Proyect. Computer Hostory Museum. Visitado en Julio de 2012: http://pdp-1.computerhistory.org/pdp-1/.

- PONG-Story. The site of the firt video game. Visitado en Julio de 2012: http://www.pong-story.com/intro.htm.

- PONG-Story. The site of the firt video game. 1952. Visitado en Julio de 2012: http://www.pong-story.com/1952.htm.

- SimsPedia. Los Sims. Visitado en Septiembre de 2012: http://es.sims.wikia.com/wiki/Los_Sims.

- The EDSAC Replica Project. Visitado en Agosto de 2012: http://www.edsac.org/.

- Vadejuegos. Videojuegos y Noticias. Multiplatazaorma y otros. Descubriendo a WillWright, el fundador del imperio Sims. Visitado en Septiembre de 2012: http://www.vadejuegos.com/noticias/2011/10/26/descubriendo-a-will-wright-el-fundador-del-imperio-sims-093209.html. [Vadejuegos] número 3 al pie de la página 23 .

- VX Vida Extra. El padre de los videojuegos. William Higinbotham, su historia, su leyenda. Visitado en Julio de 2012: http://www.vidaextra.com/industria/el-padre-de-los-videojuegos-william-higinbotham-su-historia-su-leyenda.

Fotos ordenadas por número de foto dentro del documento:

- 1. Diagrama: lanzamiento de misiles 1947. Visitado en Julio de 2012: http://www.pong-story.com/intro.htm.

- 2. Interfaz “OXO”. Visitado en Julio de 2012: http://cdn.dipity.com/uploads/events/144a57be7ed707cd9e8ae1576518825c_1M.png.

- 3. Osciloscopio en el que se ejecutaba el juego “Tennis for Two”. Visitado en Julio de 2012: http://www.vidaextra.com/industria/el-padre-de-los-videojuegos-william-higinbotham-su-historia-su-leyenda.

- 4. Pantalla del juego “Tennis for Two”. Visitado en Julio de 2012: http://cdn.dipity.com/uploads/events/ef9621cc936789207443b623a7f8ab5e_1M.png.

- 5. Pantalla del juego “Spacewar”. Visitado en Agosto de 2012: http://www.pong-story.com/intro.htm.

- 6. “ El Ajedrecista”. Visitado en Junio de 2012: http://history-computer.com/Dreamers/Torres_chess.html.

- 7. “Deep Blue” contra Gary Kasparov. Visitado en Junio de 2012: jeopardy-watson-computer-winning-gary-kasparov-deep-blue-chess_32328_600x450.jpg.

- 8. Capturas del video de presentación de "Meet Milo". Visitado en Agosto de 2012: http://www.ingamemagazine.com/tag/project-natal/ y http://www.bbc.co.uk/news/10623423.

- 10. Esquema MiniMax juego OXO sacado de los apuntes de Inteligencia Artificial Universidad de Valladolid 2007.

- 11. Ejemplo poda alfa-beta sacado de los apuntes de Inteligencia Artificial Universidad de Valladolid 2007.

- 12. Ejemplo A* sacado de los apuntes de Inteligencia Artificial Grupo PLG UC3M IA 2008.

- 13. Los Sims. Visitado en Septiembre de 2012: http://juegosabiertos.com/sim/imagenes/portadas/thesims.jpg.

- 14. Will Wright con una de sus creaciones. Visitado en Septiembre de 2012:http://www.vadejuegos.com/imagenes/2011/10/26/wright9.jpg.

- 15. 1º Expansión del juegos Los Sims. Visitado en Septiembre de 2012:http://www.gamepcrip.com/wp-content/uploads/simsmasvivosquenuncaportada.jpg.

- 16. Capturas del juego "Black&White". Visitado en Agosto de 2012: http://www.macuarium.com/macuarium/actual/especiales/2002_06_17_bw.shtml.

- 17. "Fable": ejemplo de las posibles evoluciones del personaje. Visitado en Agosto de 2012: http://espadasydados.blogspot.com.es/2011/06/fable-lost-chapters.html.

Video:

- GAME HEADZ. Discovery Chanel. Visitado en Octubre de 2012: http://www.dailymotion.com/video/xk8d2_historia-de-los-videojuegos_videogames.

10. Anexos

En este apartado se recogen los documentos que se han elaborado para el juego AI-Live. Estos documentos en vez de encontrarse dentro de esta memoria se encuentran en el repositorio que se utiliza para albergar el juego. Ambos documentos se encuentran tanto en versión pdf, como en versión .doc para que puedan ser editados en futuras versiones:

- **Manual de Usuario**: este documento explica las premisas y las indicaciones que se ha de seguir para instalar, compilar y ejecutar los diferentes módulos del juego. El manual se encuentra ubicado en el siguiente enlace: http://pruebaailivesvn.googlecode.com/files/Manual_de_usuario.pdf

- **Manual de Referencia**: este documento ayuda al entendimiento del juego AI-Live ya. que en él se explica con detalle todos las reglas, métodos, funciones e instancias de las que se encuentra formado el juego AI-Live. Se encuentra ubicado en el siguiente enlace: http://pruebaailivesvn.googlecode.com/files/Manual_de_referencia.pdf

- **Repositorio de AI-Live**: este documento ayuda al manejo de la herramienta de Google Code encargada de albergar el juego AI-Live. El documento explica como poder realizar acciones sobre el juego AI-Live como subir modificaciones o bajar las diferentes versiones para los tres sistemas operativos más populares, Windows, Linux y Mac. Se encuentra ubicado en el siguiente enlace: http://pruebaailivesvn.googlecode.com/files/Repositorio_de_AI_LIVE.pdf

- **Ontología**: este documento alberga de una forma gráfica la ontología que utiliza el juego AI-Live. Se encuentra ubicado en el siguiente enlace: http://pruebaailivesvn.googlecode.com/files/Ontologia.pdf