Sistemas Inteligentes para Mundos Virtuales

Memoria del Proyecto

Estudiante: Pilar Vizcaíno Recio

Ingeniería Informática. 2º ciclo: Proyecto Fin de Carrera

Consultora: María Antonia Huertas Sánchez

Curso 2011-2012 – Primer Semestre

AGRADECIMIENTOS:

Con este proyecto se culmina todos mis esfuerzos realizados en los últimos años con el segundo ciclo de la Ingeniería en Informática.

Cuando la carrera no es a tiempo completo sino que es similar a un pluriempleo, hay momentos en los que una se cuestiona la utilidad, el beneficio que te aporta y si es proporcional el esfuerzo que te supone. El compartir todo esto con una persona íntimamente ligado a mi como es mi hermano Jose, me ha ayudado a superar esos momentos de debilidad.

Pero mis agradecimientos son prácticamente en su totalidad a mis hijas, Neith y Alba. Ellas siempre han aceptado renunciar a ese porcentaje que les robaba mis estudios, del tiempo de estar con ellas, es más, siempre me han apoyado y animado a realizar esos esfuerzos, porque saben lo importante que es para mi.

Pero también quiero agradecer a mis padres que también han sufrido una parte de ese tiempo que les he robado en atenderlos, y al resto de mis hermanos, cuñadas, cuñado, y sobrinos que también me han apoyado en el empeño.

No quiero olvidar a mis dos amigas más íntimas, M. Angeles y M. Carmen, que están deseando que termine, por seguir ahí.

Por último, a quien más me ha animado estos últimos meses, mi consultora. Que me ha dado la oportunidad de realizar este proyecto sobre un tema tan interesante para mi "Sistemas inteligentes para mundos virtuales".

La frase "Hay otros mundos..." parece hacerse realidad con la informática y las comunicaciones, por la proliferación a pasos agigantados de entornos virtuales que nos hacen vivir otras realidades que hace años, de niños, sólo se podía imaginar como algo de la ciencia ficción. Pensar en que podemos relacionarnos en esos entornos con personajes o entidades no humanas sin que nos demos cuenta, actualmente es aún parte de esa ciencia ficción, pero no estamos tan lejos de conseguirlo.

Siempre he sido amante de la ciencia ficción y la inteligencia artificial y la realización de este proyecto me ha dado la oportunidad de conocer, en que punto estamos de conseguir que se haga realidad esa ciencia ficción, y de desear involucrarme en ello. Me hubiera gustado tener más tiempo para desarrollar el proyecto aún más, pero hay unas fechas que marcan el fin de mi prioridad en estos momentos.

Mi agradecimiento también, a todo el que lea este trabajo, que espero le resulte tan interesante como a mi.

RESUMEN:

El desarrollo de Mundos Virtuales Inteligentes requiere el conocimiento de áreas tan diversas como la realidad virtual, la inteligencia artificial, la psicología, la sociología y la física. El documento que se desarrolla a continuación recoge las nociones básicas para entender lo que representa dicho desarrollo, el estado del arte de varias de las técnicas y modelos utilizados en algunas de estas áreas y sus posibles aplicaciones, además de una posible solución para su implementación.

RESUM:

El desenvolupament de Mons Virtual Intel·ligents requeix el coneixement d'àrees tan diverses com la realitat virtual, la intel·ligència artificial, la psicologia, la sociologia i la física. El document que es desenvolupa a continuació recull les nocions bàsiques per entendre el que representa aquest desenvolupament, l'estat de l'art de les diverses tècniques i models utilitzats en algunes d'aquestes àrees i les seves possibles aplicacions, a més d'una possible solució per a la seva implementació.

ABSTRACT:

The Intelligent Virtual Worlds development requires knowledge of diverse areas of study like virtual reality, artificial intelligence, psychology, sociology and physics. The document developed below shows the basics contraits to understand what accounts for this development, the state of art of various techniques and models used in some of these areas, and their posible appications as well as a possible solution for their implementation.

PALABRAS CLAVE:

Inteligencia artificial, inteligencia emocional, entorno virtual inteligente, realidad virtual, vida artificial, avatar, actores virtuales, comportamiento emocional, lógica difusa, agente virtual emocional, humanos virtuales autónomos.

ÁREA DEL PROYECTO:

Este proyecto está desarrollado dentro del área de Representación del conocimiento y el Razonamiento.

Este área se ocupa de como expresar formalmente la información y hacer inferencias con ella. El objetivo principal en este área es encontrar un sistema formal, que describa los elementos clave del dominio que se va a representar, y que proporcione las herramientas de inferencia necesarias.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

1.1. Justificación del PFC y contexto en el cual se desarrolla: punto de partida y aportación del PFC.

Los primeros mundos virtuales aparecieron en los años 70, inicialmente con fines profesionales de aprendizaje (simuladores de vuelo[1]), de enseñanza (Massive Multiple On Line Education, MMOLE) o de medicina, pero en la actualidad están siendo utilizados con fines de ocio electrónico, en la construcción de videojuegos. Muchos de estos mundos virtuales son conocidos como videojuegos masivos en línea o MMO (Massively multiplayer online games). Más recientemente han aparecido mundos virtuales donde el usuario puede definir un alter-ego o *avatar* para interactuar en el mundo virtual (como por ejemplo *second life*). Una particularidad de todos ellos es la interacción en tiempo real.

Desde un punto de vista técnico, un mundo virtual es la combinación de un entorno gráfico 3D que incorpora sistemas de interacción social basados en chat desarrollados en el Mundo de Dominios Multi-usuario (MUDs).

Por otro lado, los sistemas basados en el conocimiento y sistemas inteligentes son los sistemas basados en reglas, dentro de los cuales destacan los sistemas expertos y los sistemas apoyados en lógicas temporales y para el razonamiento aproximado (lógicas difusas, lógicas fuzzy, etc.). Estos permiten la representación del conocimiento de un mundo y la inferencia o comportamiento inteligente.

El artículo, "*Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments"[1],* describe la convergencia entre Realidad Virtual y Inteligencia Artificial y las áreas tecnológicas que subyacen para conseguir Entornos Virtuales Inteligente, utilizando agentes virtuales con características como la autonomía, y comportamientos inteligentes como las emociones y la comunicación no verbal.

En este proyecto se estudiará como implementar un sistema inteligente inmerso en un entorno virtual, capaz de reaccionar "física" y "emocionalmente" a los estímulos de su entorno de forma autónoma, basándose en las técnicas de Inteligencia Artificial y Realidad Virtual.

1.2. Objetivos del PFC.

El objeto del PFC es analizar las posibilidades de creación de sistemas inteligentes en mundos virtuales, especialmente contemplando el factor temporal.

Esto implica dos objetivos claros:

- Estudiar la posibilidad actual de creación de estos sistemas en mundos virtuales, con especial atención a los factores temporales.
- Diseñar un sistema inteligente de estas características.

1 En 1979 se crea el primer simulador de vuelo basado únicamente en sistemas informáticos.

Para alcanzar el primer objetivo, la posibilidad de creación de estos sistemas inteligentes en un mundo virtual, será necesario realizar un trabajo de investigación sobre estado del arte de las técnicas y herramientas en los campos de realidad virtual (VR) y inteligencia artificial (IA).

Para el segundo objetivo, se realizará una simulación de un entorno virtual que incluirá un agente virtual autónomo (AVA) dotado de un comportamiento inteligente mediante un algoritmo de inferencia.

El problema se centrará en uno de los aspectos considerados como comportamiento inteligente que está más en auge en la actualidad; la comunicación no verbal.

La comunicación no verbal responde principalmente a la expresión de las emociones, que se reflejan en el rostro, en el cuerpo y en la entonación al hablar.

Por un lado se están haciendo grandes avances, en el campo de la realidad virtual, respecto a la comunicación no verbal para dar mayor realismo a actores sintéticos.

Por otro lado, en Inteligencia Artificial, se han empezado a dar los primeros pasos con proyectos como el desarrollo de sistemas capaces de "comprender" la emoción humana mediante la utilización de una gran base de datos de patrones de la expresión emocional de los seres humanos.

Según datos de un artículo en la web de *"Noticias actuales sobre tecnología e informática"*[2] se ha desarrollado un software capaz de determinar si una persona está alegre, triste, sorprendida o rabiosa, en tiempo real, con un alto grado de exactitud, por ejemplo, el grado de exactitud en determinar si quien hay en frente es un hombre o una mujer es del 90%.

El problema que se pretende resolver en este proyecto, es el de representar un sistema capaz de "sentir" una emoción dependiendo de una serie de estímulos externos.

Se trataría de que el sistema AVA recogiera información de un escenario diseñado a tal efecto, sobre aspectos cambiantes con el tiempo, que influirían en el "estado de ánimo" del agente y como resultado hiciera cambiar su "expresión facial". Para delimitar aún más el problema, el diseño se centrará en una emoción en concreto; la alegría, y uno o varios estímulos cambiantes.

Los estímulos que se tendrán en cuenta serán por un lado estímulos físicos; el escenario que representará el tiempo (estación del año, condiciones meteorológicas, etc.) y por otro lado estímulos sociales, sentirse rodeado de gente alegre (no demasiada) , el reencuentro con "amigos", etc.

El entorno, los estímulos cambiantes y la reacción emocional del AVA se desarrollará en VRML.

Como el proyecto se encuentra encuadrado dentro del área de la Representación del Conocimiento y el Razonamiento, la implementación del problema se centrará principalmente en el modelado de la obtención de los estímulos externos y de la base de conocimiento, y del algoritmo de inferencia, dejando en un segundo plano la representación gráfica, que se realizará mediante formas geométricas simples que emulen a la gente y al propio "agente inteligente" y con colores que simularán el grado de alegría del agente.

1.3. Enfoque y método seguido.

Este proyecto tiene dos objetivos claramente diferenciados que suponen cada uno un enfoque y método propio.

Para el primer objetivo claramente teórico (estudiar la posibilidad de creación de un sistema inteligente en un mundo virtual), se ha realizado una búsqueda exhaustiva por Internet de todos los conceptos relacionados con tema del proyecto, de las herramientas, modelos que se utilizan, investigaciones en curso, y artículos de opinión, para acabar proponiendo una posible solución genérica utilizando unas determinadas herramientas, que no tiene porque ser la mejor solución, ya que, esta extensa investigación no ha permitido profundizar con la misma intensidad en todos los aspectos, ni tampoco aseguran el conocimiento de la totalidad de las herramientas posibles.

Para el segundo objetivo "implementar un sistema inteligente inmerso en un entorno virtual", se ha seguido un enfoque más práctico, con el estudio de las herramientas seleccionadas ya en la descripción del problema, que ha supuesto en primer lugar un estudio y práctica del lenguaje para el desarrollo de entornos virtuales VRML, el lenguaje de control de lógica difusa FCL y de las librerías de Java para obtener la información del entorno y para poder realizar la inferencia. En concreto varias librerías de EAI para VRML y la librería JfuzzyLogic.

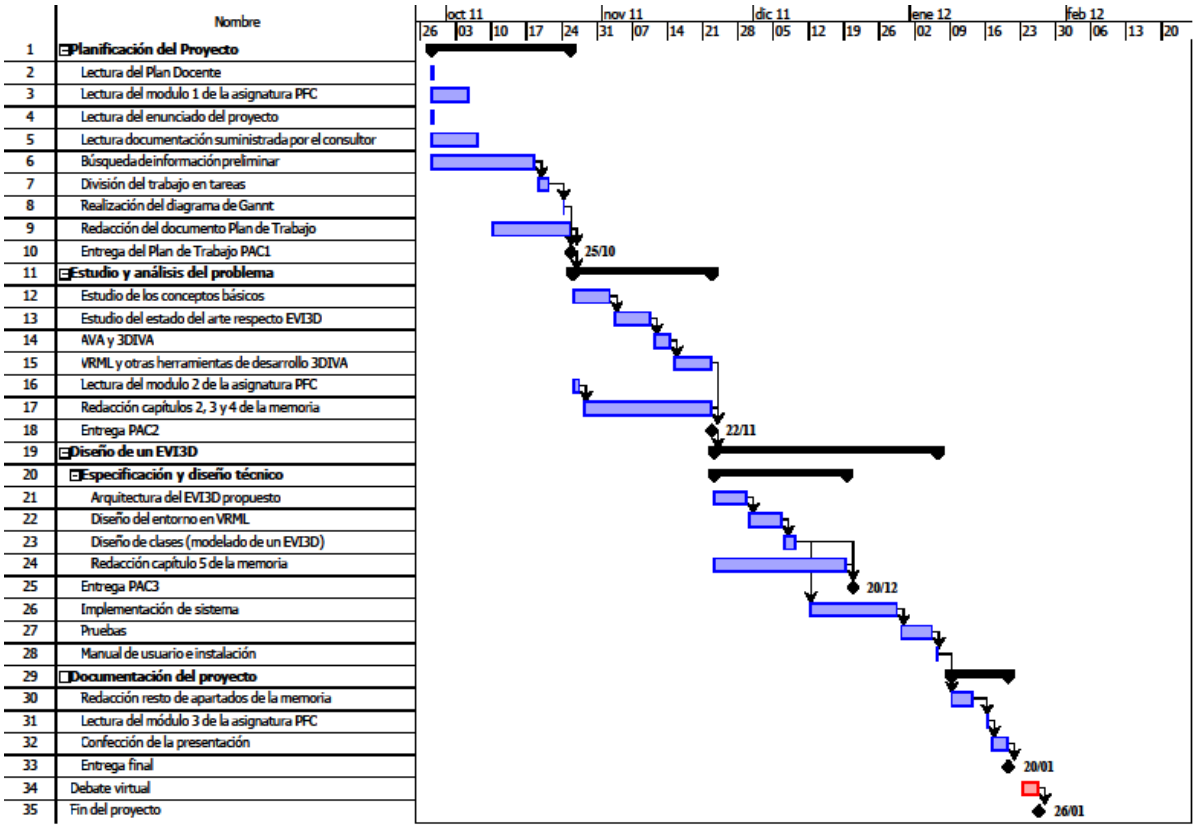
1.4. Planificación del proyecto. Hitos y temporalización.

1.4.1. Temporalización:

Se ha realizado una temporalización ajustada a los hitos que son las diferentes entregas a realizar, puede sufrir alguna variación durante el desarrollo del proyecto y por ello en cada entrega se acompañará un documento de seguimiento de la planificación con los ajustes que se estimen oportunos.

	Nombre	Duración	Inicio	Terminado	Predecesores
1	**Planificación del Proyecto**	**20 days?**	**28/09/11 8:00**	**25/10/11 17:00**	
2	Lectura del Plan Docente	1 day?	28/09/11 8:00	28/09/11 17:00	
3	Lectura del modulo 1 de la asignatura PFC	6 days?	28/09/11 8:00	5/10/11 17:00	
4	Lectura del enunciado del proyecto	1 day?	28/09/11 8:00	28/09/11 17:00	
5	Lectura documentación suministrada por el co...	8 days?	28/09/11 8:00	7/10/11 17:00	
6	Búsqueda de información preliminar	15 days?	28/09/11 8:00	18/10/11 17:00	
7	División del trabajo en tareas	3 days?	19/10/11 8:00	21/10/11 17:00	6
8	Realización del diagrama de Gannt	1 day?	24/10/11 8:00	24/10/11 17:00	7
9	Redacción del documento Plan de Trabajo	12 days?	10/10/11 8:00	25/10/11 17:00	
10	Entrega del Plan de Trabajo PAC1	0 days?	25/10/11 17:00	25/10/11 17:00	8;9
11	**Estudio y análisis del problema**	**20 days?**	**26/10/11 8:00**	**22/11/11 17:00**	**10**
12	Estudio de los conceptos básicos	6 days?	26/10/11 8:00	2/11/11 17:00	
13	Estudio del estado del arte respecto EVI3D	6 days?	3/11/11 8:00	10/11/11 17:00	12
14	AVA y 3DIVA	2 days?	11/11/11 8:00	14/11/11 17:00	13
15	VRML y otras herramientas de desarrollo 3DIVA	6 days?	15/11/11 8:00	22/11/11 17:00	14
16	Lectura del modulo 2 de la asignatura PFC	2 days?	26/10/11 8:00	27/10/11 17:00	
17	Redacción capítulos 2, 3 y 4 de la memoria	18 days	28/10/11 8:00	22/11/11 17:00	16
18	Entrega PAC2	0 days?	22/11/11 17:00	22/11/11 17:00	15;17
19	**Diseño de un EVI3D**	**33 days?**	**23/11/11 8:00**	**6/01/12 17:00**	**18**
20	**Especificación y diseño técnico**	**19 days?**	**23/11/11 8:00**	**19/12/11 17:00**	
21	Arquitectura del EVI3D propuesto	5 days?	23/11/11 8:00	29/11/11 17:00	
22	Diseño del entorno en VRML	5 days?	30/11/11 8:00	6/12/11 17:00	21
23	Diseño de clases (modelado de un EVI3D)	3 days?	7/12/11 8:00	9/12/11 17:00	22
24	Redacción capítulo 5 de la memoria	19 days?	23/11/11 8:00	19/12/11 17:00	
25	Entrega PAC3	0 days?	20/12/11 17:00	20/12/11 17:00	23;24
26	Implementación de sistema	14 days?	12/12/11 8:00	29/12/11 17:00	23
27	Pruebas	5 days?	30/12/11 8:00	5/01/12 17:00	26
28	Manual de usuario e instalación	1 day?	6/01/12 8:00	6/01/12 17:00	27
29	**Documentación del proyecto**	**10 days?**	**9/01/12 8:00**	**20/01/12 17:00**	
30	Redacción resto de apartados de la memoria	5 days?	9/01/12 8:00	13/01/12 17:00	28
31	Lectura del módulo 3 de la asignatura PFC	1 day?	16/01/12 8:00	16/01/12 17:00	30
32	Confección de la presentación	4 days?	17/01/12 8:00	20/01/12 17:00	31
33	Entrega final	0 days?	20/01/12 17:00	20/01/12 17:00	32
34	Debate virtual	4 days?	23/01/12 8:00	26/01/12 17:00	
35	Fin del proyecto	0 days?	26/01/12 17:00	26/01/12 17:00	34

1.4.2. Diagrama de Gantt:



1.4.3. Hitos:

- 25-10-2011. Presentación de la PAC1: Plan de Trabajo.
- 22-11-2011. Presentación de la PAC2: Capítulos 2, 3, y 4 de la memoria y seguimiento de la planificación.
- 20-12-2011. Presentación de la PAC3: Capítulo 5 de la memoria y seguimiento de la planificación.
- 20-01-2012. Entrega final: Memoria completa, presentación en powerpoint, fuentes y ejecutables del proyecto y manuales de usuario y de instalación.
- 26-01-2012. Fin del proyecto.

1.5. Productos obtenidos.

Del trabajo realizado durante el desarrollo de este proyecto han resultado los siguientes productos:

- La memoria del proyecto (este documento) dónde se expone todo el trabajo realizado de investigación y los resultados obtenidos.

- Una serie de ficheros con extensión wrl que son el entorno virtual dónde se aplica el sistema de inferencia desarrollado..
- Un fichero con extensión fcl que es el sistema de inferencia utilizado.
- Las clases de Java para obtener la información del entorno, utilizar el sistema de inferencia desarrollado y mostrar los resultados de este.
- Un manual de instalación (anexo 1 de este documento) para instalar los programas necesarios para poder probar el sistema de inferencia.
- Un manual de usuario (anexo 2 de este documento) con las instrucciones para poder probar el funcionamiento del sistema de inferencia desarrollado y del entorno virtual.
- Y una fichero con extensión pps que es la presentación en powerpoint de síntesis del proyecto.

1.6. Breve descripción del resto de capítulos de la memoria.

- En el capítulo 2 “Conceptos generales” se desarrollan los conceptos utilizados durante el proyecto, es a la vez una visión del trabajo inicial de investigación realizado.

  Comienza con los conceptos de realidad virtual y mundos virtuales que incluye algo de historia de su desarrollo. El concepto de vida artificial que introduce comportamiento en entornos virtuales. Para terminar con las bases de la lógica difusa y las teorías y modelos computacionales para representar e inferir emociones.
- En el capítulo 3 “Estudio de los actuales Entornos Virtuales y las herramientas que los soportan” se introducen algunos ejemplos significativos actuales de aplicación de la realidad virtual, entornos como SecondLife y videojuegos, y herramientas para el modelado y la visualización de entornos virtuales.
- En el capítulo 4 “Análisis de la posibilidad de creación de Entornos Virtuales Inteligentes en la actualidad, contemplando el factor temporal” se concretan las herramientas necesarias para la creación de un posible entorno virtual inteligente utilizando como base un modelo de representación de las emociones con lógica difusa como es FLAME.
- En el capítulo 5 “Diseño de un Entorno Virtual Inteligente” describe el análisis y diseño del problema objeto de este PFC, que supone la realización de un prototipo simple de un entorno virtual y un sistema de inferencia que se nutre de los datos que se obtienen de dicho entorno.
- Por último, en el capítulo 6 “Conclusiones y lineas futuras” se tratan las limitaciones del prototipo construido y de las posibilidades de desarrollo futuro y sus aplicaciones.

2. CONCEPTOS GENERALES

2.1. Realidad virtual y Mundos virtuales.

El término Realidad Virtual puede resultar a primera vista paradójico al tratarse se un término compuesto de dos conceptos prácticamente opuestos. La realidad es algo tangible que tiene existencia verdadera y efectiva, y lo virtual es todo lo contrario, es algo que no existe físicamente, algo incorpóreo. Por ese motivo algunos investigadores tienden a utilizar otros términos como: Realidad Artificial o Ambientes sintéticos.

Sin embargo, Realidad Virtual y Realidad Artificial no son sinónimos. El Término Realidad Virtual se utiliza para describir simulaciones visuales de la realidad física que nos rodea. La Realidad Artificial simula entornos y escenas inexistentes, o imposibles porque incumplen leyes físicas.

Seria, pues, más adecuado asimilar el término Realidad Virtual con el término Ambientes Sintéticos, como la simulación de medios ambientes y de los mecanismos sensoriales del hombre por computadora, de tal manera que proporcione al usuario la sensación de inmersión y la capacidad de interacción con medios ambientes artificiales.

Aún así, la tecnología de la Realidad Virtual asume todo lo que puede ofrecer la realidad artificial, todo lo que permite la imaginación, mezclando lo artificial con lo real.

Con todo ello, se podría definir la Realidad Virtual como una simulación tridimensional e interactiva generada por computador en la que el usuario se siente introducido en un ambiente artificial, y que lo percibe como real basado en estímulos de los órganos sensoriales. Una realidad ilusoria, ya que se trata de una realidad perceptiva sin soporte físico, pues existe sólo dentro del ordenador. La realidad Virtual es, de esta manera, el medio que proporciona una visualización interactiva en tres dimensiones para simular mundos virtuales.

El término mundo virtual se puede definir como un tipo de comunidad virtual en linea que simula un mundo o entorno artificial, inspirado o no en la realidad, en el cual los usuarios pueden interactuar entre sí a través de personajes o avatares, y usar objetos o bienes virtuales.

Para ser un mundo virtual, se requiere un mundo en línea persistente, activo y disponible 24 horas al día y todos los días, para que los usuarios vivan e interactúen, generalmente en tiempo real.

Durante algún tiempo se asimiló el término de mundo virtual al de Metaverso, Sin embargo el término metaverso es más amplio y engloba a cuatro tipos distintos de mundos sintéticos: Juegos y mundos virtuales, Mundos espejos (Google Earth), Realidad aumentada y Lifelogging.

Los mundos o entornos virtuales pueden ser de dos tipos: inmersivos y no inmersivos, dependiendo de los dispositivos adicionales que se utilicen. En mundos virtuales inmersivos el usuario desconecta todos sus sentidos del mundo real mientras que en los no inmersivos la desconexión no es total.

En el apartado 2.1.4 de este capítulo se relacionan diferentes dispositivos hardware que se utilizan en

Realidad Virtual inmersiva. La Realidad Virtual no inmersiva o también llamada semiinmersiva utiliza medios como el que actualmente nos ofrece internet con el cual se puede interactuar en tiempo real con otras personas en espacios y ambientes que en realidad no existen, sin la necesidad de dispositivos adicionales.

2.1.1. Un poco de historia.

El concepto de Realidad Virtual surge en 1965 cuando Ivan Sutherland (hoy miembro de Sun Microsystems Laboratories), publicó un artículo titulado "The Ultimate Display", en el cual describía el concepto básico de la Realidad Virtual.

Al año siguiente Sutherland creó el primer casco visor de Realidad Virtual, era un instrumento bastante primitivo que fue llamado "Espada de Damocles", debido a que el aparato requería de un sistema de apoyo que pendía del techo. El funcionamiento del sistema era el siguiente. Un brazo mecánico articulado, fijado al techo, sostenía un sistema de visualización compuesto por dos pequeñas pantallas (CRT) con un soporte para ser ajustado a la cabeza de un usuario. Las articulaciones del brazo estaban dotadas de potenciómetros que medían los cambios de orientación de la cabeza del usuario. Un ordenador generaba pares estereoscópicos de imágenes de objetos en tres dimensiones representados en formato de alambre (*wireframe*) mediante una proyección en perspectiva, las cuales eran enviadas a las pantallas del sistema de visualización permitiendo al usuario ver los objetos. Los movimientos del usuario, detectados por los sensores de las articulaciones del brazo, eran enviados al ordenador con el objetivo de modificar la orientación del punto de vista dentro del entorno geométrico definido por los objetos.

Ilustración 1: Espada de Damocles de Sutherland

Dos años después en 1968, junto con David Evans, Sutherland crea el primer generador de escenarios con imágenes tridimensionales, datos almacenados y aceleradores.

Los simuladores de vuelo han sido una importante vía de desarrollo para la Realidad Virtual, y es en 1972 cuando General Electric, desarrolla el primer simulador computarizado de vuelo para la Armada Norteamericana.

El primer sistema que se proponía atravesar el umbral de las dos dimensiones y recrear una verdadera "Telepresencia en un espacio de datos" fue concebido en los laboratorios de la NASA. El proyecto Virtual Environment Display, iniciado en 1984, tenía como objetivo la construcción de una estación de trabajo virtual destinada a ser utilizada en misiones espaciales de la NASA.

Con la creación de los dispositivos hardware cada vez más sofisticados, como los guantes de manipulación o los trajes sensitivos completos, con sensores y visores especiales, la Realidad Virtual involucra además de la vista y el oído, el tacto y el llamado "sentido del cuerpo".

Toda esta diversidad de dispositivos han hecho que los campos de aplicación no se limiten al ocio, a pesar de que la aplicación en ocio y entretenimiento ha contribuido a disparar el interés en el sector privado para invertir en investigación.

2.1.2. Características de un entorno virtual.

- Simulación: capacidad de replicar aspectos suficientes de un objeto o ambiente de forma que pueda convencer al usuario de su casi realidad.
- Tridimensionalidad: Tiene que ver directamente con la estimulación de los sentidos del usuario, principalmente la visión, para dar forma al espacio virtual. Los componentes del mundo virtual se muestran al usuario en las tres dimensiones del mundo real. En lo que se refiere a los sonidos se tienen efectos estereofónicos.
- La inmersión: o mejor la sensación de inmersión, es una propiedad mediante la cual el usuario tiene la sensación de encontrarse dentro de un mundo tridimensional.
- Existencia de un punto de observación o referencia: permite determinar ubicación y posición de observación del usuario dentro del mundo virtual.
- Navegación: propiedad que permite al usuario cambiar su posición de observación.
- Manipulación: característica que posibilita la interacción y transformación del medio ambiente virtual.

2.1.3. Componentes

Existen dos tipos fundamentales de componentes de la Realidad Virtual. Los elementos Hardware, que son elementos externos que permiten la interacción con el mundo virtual. Y por otro lado los elementos Software que generan el propio mundo virtual.

Elementos Hardware:

Dependiendo de la complejidad del sistema de Realidad Virtual los dispositivos serán más o menos complejos.

Dispositivos de entrada: Con los que el usuario podrá transmitir sus órdenes al sistema de realidad virtual, indicándole que desea desplazarse o cambiar el punto de vista o interactuar con algún objeto: Guantes y trajes de datos, Joysticks 3D, Mouse 3D, Track Balls, Rastreadores de posición y movimiento, además de los básicos, como teclado, mouse, micrófono y webcam.

Ilustración 2: Casco y guantes Virtual

Dispositivos de salida: Permiten que el usuario se sienta inmerso en el mundo virtual creado: Cascos visores, Sistemas binoculares, Lentes estereoscópicos, Sonido tridimensional y por supuesto la pantalla y los altavoces.

Elementos Software:

Los programas de software para el desarrollo de aplicaciones de Realidad Virtual se fundamentan en los conceptos de computación gráfica. Las características fundamentales son:

- Capacidad para importar de modelos 3D y librerías de objetos 3D
- Permitir operaciones geométricas
- Poder distinguir en el nivel de detalle de los objetos más cercanos frente a los más lejanos.
- Animación sincronizada con la navegación por el ambiente virtual.
- Objetos organizados por jerarquías
- Detección de colisiones entre objetos
- Propiedades físicas de los objetos
- Color, texturización y fuentes de luz
- Permitir la subdivisión del ambiente virtual en submundos
- Conectividad en red para poder ser utilizado por varios usuarios

Algunos ejemplos de estos programas son: VRML, 3D Studio Max, V-Realm, Multigen, RayDream Studio.

2.1.4. Aplicaciones de la Realidad Virtual.

- Medicina: Telecirugía, Microcirugía. En muchos casos se utilizan robots controlados por Telepresencia
- Educación: El campo de la educación es uno de los mas beneficiados de los avances en la realidad virtual. Actualmente existen experiencias de distintos grupos de investigación para la creación de material educativo.
- Marketing y Comercio Electrónico: Visita de tiendas virtuales.
- Defensa: Permitiendo realizar simulaciones en medios hostiles, como por ejemplo reparaciones en el interior de un reactor nuclear.
- Arquitectura: Los arquitectos pueden hacer que sus clientes visiten los pisos-piloto en un mundo de Realidad Virtual, dándoles la oportunidad de abrir puertas o ventanas, etc. Además permite la anticipación de errores de diseño.
- Ocio y entretenimiento: Videojuegos tanto individuales (estaría dentro de la Realidad Virtual de Escritorio) como multiusuario (Realidad Virtual en segunda persona), juegos de rol, simuladores y redes sociales. Por la naturaleza de la realidad virtual, en el cine es un medio ideal de representación de la ciencia ficción, incluso muchas veces se utiliza como escenario (Tron y muchas secuencias de la saga Matrix)

2.1.5. Parámetros para comparar sistemas de Realidad Virtual.

- Velocidad de respuesta.
- Calidad de las imágenes proyectadas.
- Números de sentidos que intervienen en la inmersión.
- Calidad con que se simulan
- Calidad con que se logran los efectos de inmersión y manipulación del ambiente virtual.

2.2. Vida Artificial.

Los entornos virtuales 3D tal como se han descrito en el apartado anterior (2.1.) son entornos que pueden llegar a adquirir un alto grado de calidad gráfica y realismo para los sentidos, pero para conseguir equiparar un escenario artificial al mundo tal como lo conocemos es necesario llenarlo de animación, con objetos y actores con los que el usuario no solo pueda interactuar sino que también tengan "vida" propia igual que en el mundo real.

La Vida Artificial trata de incorporar reglas de comportamiento, aprendizaje y evolución a modelos virtuales y robóticos. De esta manera los personajes o entidades pueden o no ser autónomos y evolucionar en su comportamiento, reproducirse y heredar y mejorar las características de su especie.

Para introducir la animación gráfica que simule "vida" en un entorno virtual 3D, será necesario un alto grado de realismo de comportamiento tanto del entorno como de los diferentes tipos de criaturas o entidades más o menos dotadas de autonomía e incluso inteligencia propia que se agreguen. Estas entidades deben ser modeladas independientes del entorno o escenario virtual, pero estarán inmersas en el mismo.

2.2.1. Avatar

Un avatar es la representación gráfica, generalmente humana, que se asocia a un usuario para su identificación. Su comportamiento es explícitamente controlado por el propio usuario externo a quien representa, mediante comandos de control, por lo tanto no es autónomo.

Los avatares pueden ser fotografías o dibujos artísticos, aunque las tecnologías en entornos virtuales en 3D suelen permitir representaciones tridimensionales.

En entornos multiusuario los avatares, a pesar de no tener vida autónoma, contribuyen a dotar de vida al mundo en el que se encuentran. La calidad con la que un avatar podrá simular una persona dependerá de las posibilidades de control para actuar y de las cualidades que se le permita definir al usuario en el entorno.

2.2.2. Objeto inteligente

Un Objeto Inteligente es un objeto virtual que es modelado con sus características de interacción; o sea, con todas las partes, movimientos y descripciones del objeto que son importantes cuando ocurre una interacción con un agente. Los Objetos inteligentes proveen los parámetros necesarios para la generación de

movimiento, así como sus propiedades intrínsecas (posición, masa y apariencia).

Un ejemplo de un objeto inteligente podría ser una pelota, con la que un actor virtual podría interactuar con ella, lanzándola con la mano o chutándola con el pie. La pelota tendrá un comportamiento adecuado a sus características y la interacción recibida. Este tipo de objeto son considerados inteligentes porque encapsulan la información necesaria para permitir interaccionar con ellos.

Las aplicaciones con objetos inteligentes proporcionan un número de ventajas superior a otros enfoques, ya que descentraliza el control de la animación, separando la animación de nivel alto de la de nivel bajo, y permitiendo al mismo objeto ser usado en múltiples aplicaciones; además de promover el diseño orientado a objetos, ya que cada objeto encapsula sus propios datos.

2.2.3. Agentes virtuales autónomos

Los Agentes Virtuales Autónomos (AVAs) son entidades de software que representan personajes virtuales inmersos en escenarios 3D, los cuales tienen la capacidad de reaccionar al entorno que los rodea, simulando tener vida propia.

Es una entidad independiente de software que, aunque solo existe y funciona en un entorno específico, es "consciente" de los cambios que se producen a su alrededor y es capaz de responder a ellos de manera autónoma, sin necesidad de instrucciones o control externo, exhibiendo gráficamente un comportamiento como el de un ser vivo.

La arquitectura de un agente virtual autónomo podría representarse de la siguiente manera[3]:

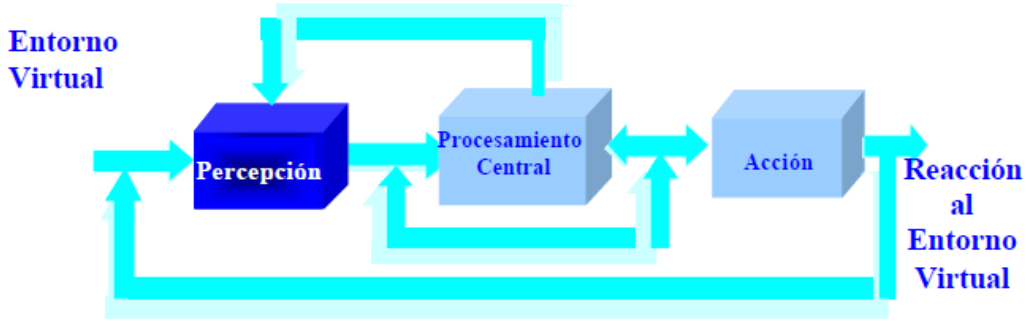


Ilustración 3: Representación del Modelo de triple torre de Nilsson

Cuando el tipo de entidad que se pretende simular es una persona algunos autores se refieren a los agentes virtuales como humanos virtuales que pueden ser o no autónomos. Por el contrario cuando un agente virtual autónomo pretende simular a animales normalmente se les denomina actores virtuales o criaturas virtuales.

Según el artículo "Intelligent agents: Theory and practice"[4], existen dos nociones de agentes virtuales autónomos una débil y otra fuerte. Las características que identifican a los agentes débiles son: Autonomía, Habilidad social, Reactividad y Pro-actividad. Si se les añaden algunas de las siguientes características: Nociones mentales, Racionalidad, Veracidad y Adaptabilidad o aprendizaje, entonces estaríamos tratando con la noción fuerte en la definición de los agentes, a los que podríamos llamar Agentes Virtuales Inteligentes.

Los Agentes Virtuales reciben información del entorno que les rodea por medio de sensores virtuales que pueden ser visuales, táctiles o auditivos. En la siguiente ilustración, obtenida del artículo "Perception and Learning in Artificial Animals"[5], se describe el modelo de comportamiento de un pez virtual.

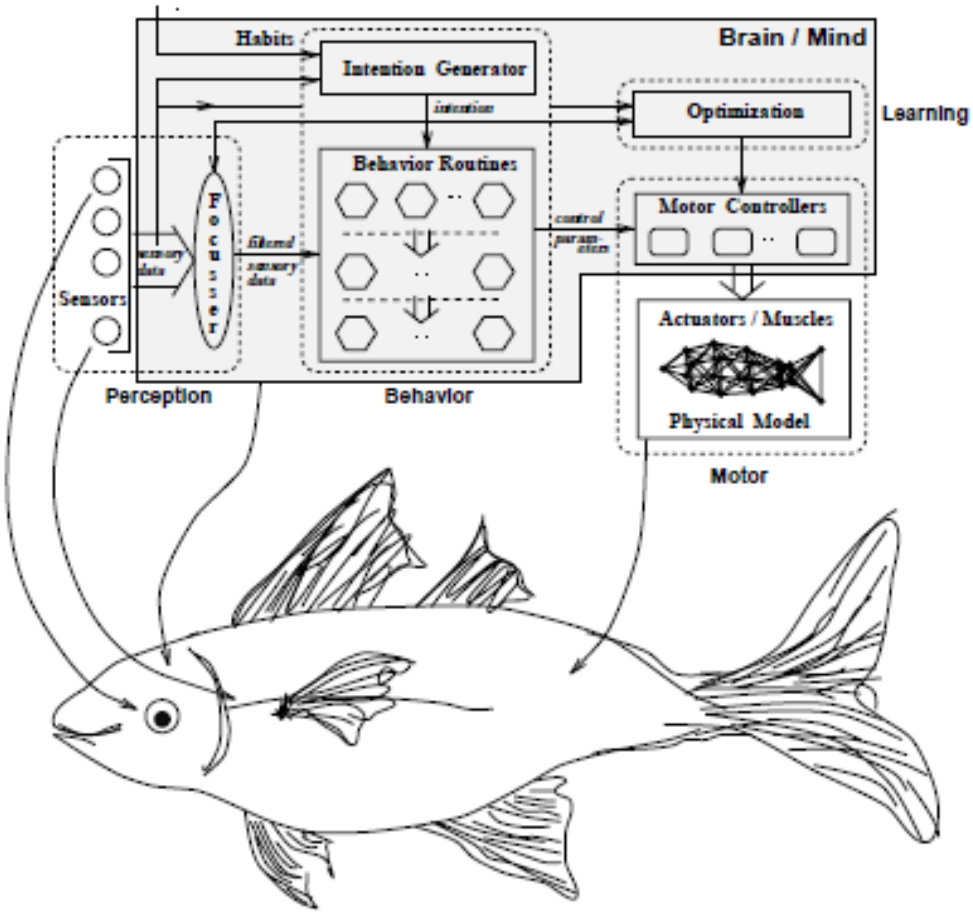


Ilustración 4: Modelo de comportamiento de Terzopoulos

En el desarrollo de Agentes Virtuales Autónomos, el mayor reto es construir humanos virtuales autónomos inteligentes. Para ser considerados inteligentes su conducta debe ir acorde a la conducta de un ser humano, es decir, debe ser capaz de actuar libre y emocionalmente, ser consciente e impredecible, y ser coherente consigo mismo y su entorno.

2.2.4. Humanos Virtuales Autónomos Inteligentes

Dentro de la categoría de Agentes Virtuales Autónomos, se encuentran los agentes que pretende simular el comportamiento lo más realista posible de una persona, tanto a nivel físico como de comportamiento e inteligencia.

Las técnicas para conseguir este tipo de comportamiento tan complejo se conjugan para formar lo que se conoce como Marco de Simulación del Comportamiento. El diseño de estos módulos de comportamiento de los AVAs son independientes del diseño del mundo virtual que habitan, y les permiten reconocer el entorno y ofrecer respuestas acordes a los estímulos, tanto externos como internos, que reciben.

En el año 2000, Ángela Caicedo y Daniel Thalmann presentaron un Marco de Simulación del Comportamiento para construir humanos virtuales[6]:

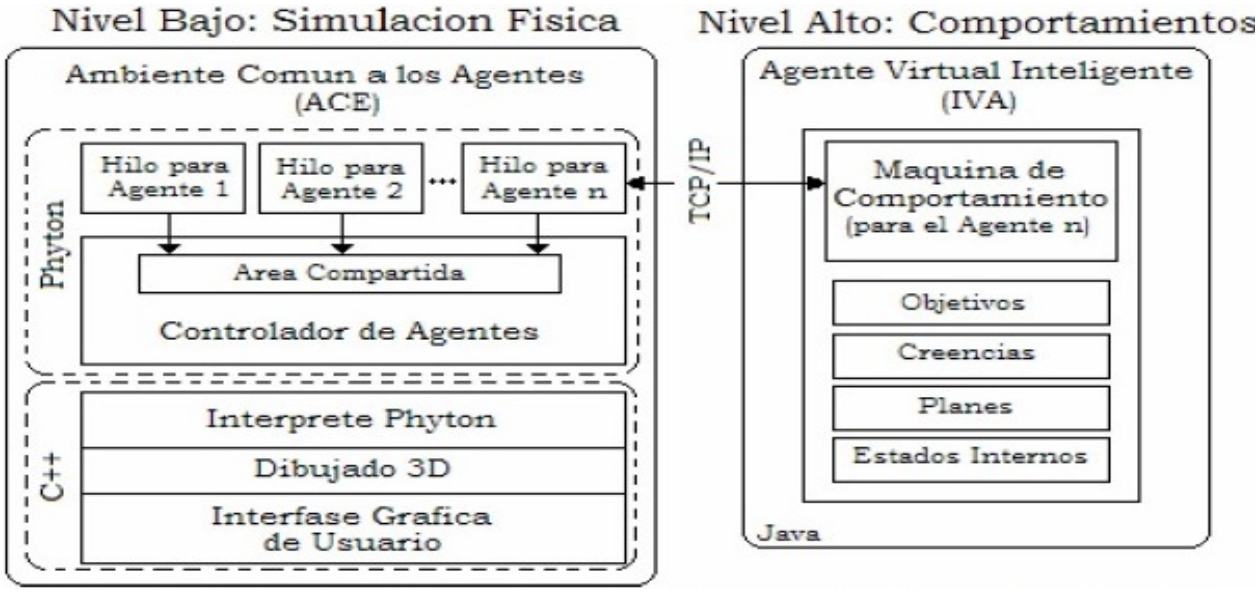


Ilustración 5: Marco de simulación del comportamiento propuesto por Thalmann y Caicedo

El Modulo de comportamiento es una Unidad de Razonamiento denominada IVA (Intelligent Virtual Agent), que se conecta al modulo de simulación física del humano.

Los comportamientos más complejos o elaborados para AVAs se encuentran en los humanos virtuales autónomos.

De la misma manera, el sistema de simulación física en humanos virtuales autónomos, también resulta mucho más complejo que en otro tipo de agentes. Un ejemplo es, la comunicación no verbal inherente en las personas. En concreto las expresiones faciales resultantes:

Modelado de gestos faciales:

Para el modelado de gestos faciales de forma eficiente, en muchos casos es necesario modelar los músculos faciales. Por otra parte la cara se divide en diferente grupos expresivos que están bien definidos: las cejas, los ojos, la nariz, la boca y el mentón. Se introducen en el modelo las líneas que definen la expresión de la cara en cada una de estas partes.

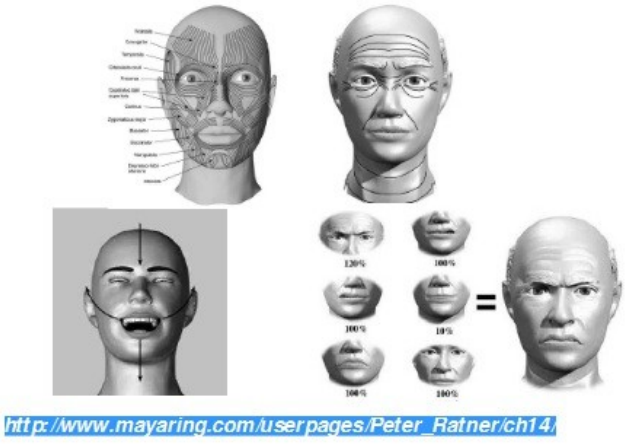


Ilustración 6: Modelado de gestos faciales

El movimiento de estas líneas junto con los músculos definirá la expresión final de la cara. Combinando

distintas expresiones es posible obtener expresiones muy realistas.

Las emociones en Humanos Virtuales Autónomos Inteligentes:

En 1998, Thalman presenta el siguiente Sistema de Simulación del Comportamiento para AVAs motivados por emociones[7]:

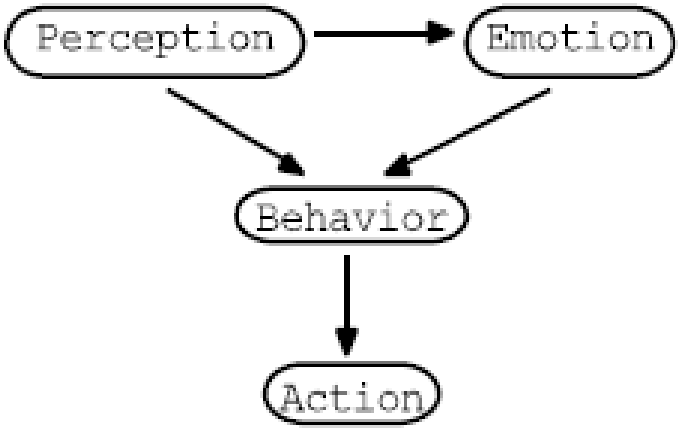


Ilustración 7: Estructura del modelo de comportamiento motivado por emociones

Según el artículo de Thalmann, en esta arquitectura el modulo percepción puede descomponerse en tres tipos: percepción de presencia de objetos y otros agentes en su entorno, percepción de las acciones de los otros agentes y percepción de los eventos que son las acciones que los agentes producen en objetos.

La percepción de eventos a su vez se descompone en eventos deseables, eventos que hacen felices a otros agentes y eventos potenciales que pueden o no ocurrir.

Las emociones que puede sentir un agentes son causadas por las percepciones, si cumplen unas condiciones y con variables que miden su intensidad.

Cada emoción es evaluada en dos pasos. Se calcula un potencial emocional a partir de un conjunto de elementos que contribuyen a la emoción y si el potencial sobrepasa un umbral dado, entonces su valor se usa para calcular la intensidad de la emoción. El umbral es la mínima intensidad del potencial para que el agente sienta la emoción.

El módulo de comportamiento es el responsable de la selección y ejecución del comportamiento que debe exhibir el agente como consecuencia de sus emociones. A su vez el módulo controla la ejecución de los comportamientos y las transiciones entre comportamientos.

El modulo de comportamiento descompone comportamiento en comportamientos más simples, que a su vez se vuelven a descomponer, formando una jerarquía de comportamientos dónde en cada nivel de la jerarquía contiene uno o más comportamientos que pueden ser ejecutados secuencialmente o concurrentemente. Los comportamientos generan la ejecución de acciones que controla el módulo de acción.

En el artículo se incluyen varios ejemplos en el que se consideran emociones como: agrado, desagrado, pena, admiración, tristeza o temor.

Existen, sin embargo, otras arquitecturas para modelar el comportamiento de los agentes virtuales inteligentes utilizando lógica difusa que se verán en el siguiente apartado.

2.3. FuzzyLogic

La lógica borrosa emerge como una herramienta interesante para el control y representación de subsistemas y procesos industriales complejos, así como también para la electrónica de entretenimiento y hogar, sistemas de diagnóstico y otros sistemas expertos. Con este sistema formal se pretende representar de forma rigurosa el significado de los enunciados imprecisos del lenguaje natural.

El aspecto central de este sistema formal radica en la modelización de modos de razonamiento imprecisos, los cuales juegan un rol esencial en la habilidad humana de trazar decisiones racionales en un ambiente de incertidumbre e imprecisión.

La lógica difusa aparece por primera vez en un artículo de Lotfi A. Zadeh publicado en 1965 y titulado "Fuzzy Sets" (Conjuntos Difusos) [“Fuzzy Sets”. Zadeh, 1965]. Pero hay que tener en cuenta que la idea de la parametrización “borrosa” de las cosas, se hereda desde la época de los primeros grandes filósofos. Posteriormente a ellos, otros grandes pensadores como David Hume o Kant apoyaban esta idea manteniendo que el razonamiento venía dado por las observaciones de las que somos testigos a lo largo de nuestra vida y la detección de algunos principios contradictorios en la lógica clásica.

Se podría definir la lógica difusa como los métodos y principios del razonamiento a partir de proposiciones imprecisas que relacionan magnitudes y valores cualitativos modelados por conjuntos difusos. Su esquema de razonamiento es el razonamiento aproximado.

Los sistemas difusos son sistemas que se basan en reglas, que se resuelven transformando los valores numéricos de entrada en valores de lógica difusa, con el motor de inferencia que utiliza las reglas y por último convirtiendo los valores de lógica difusa a valores numéricos de salida.

A continuación se realiza un repaso de algunos de los conceptos utilizados en lógica difusa.

2.3.1. Base teórica de la lógica difusa.

Conjuntos difusos.

El concepto fundamental en que se basa la lógica difusa es el de conjunto difuso, un tipo de conjunto caracterizado porque los elementos del universo de discurso en el que está definido pueden pertenecer a él en un cierto grado, representado por una función de pertenencia.

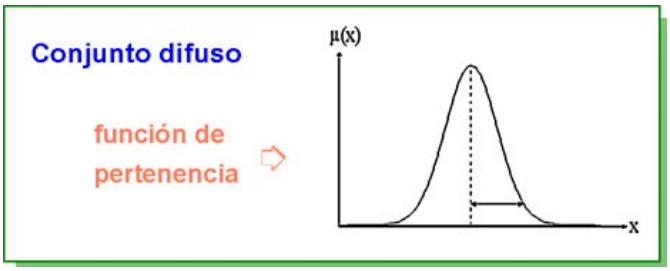


Ilustración 8: Función de pertenencia en conjuntos difusos.

Un conjunto difuso A sobre un universo de discurso U es un conjunto de pares dado por:

$$A = \{ \mu A(u) / u : u \in U, \mu A(u) \in [0,1] \}$$

Relaciones difusas:

El concepto de relación difusa es una generalización del concepto de relación de la teoría clásica de conjuntos. Mientras que una relación entre dos conjuntos clásicos describe la existencia o no de asociación entre los elementos de ambos conjuntos, una relación difusa describe el grado de asociación o interacción entre los elementos de dos o más conjuntos difusos.

La relación difusa puede representarse mediante una matriz, denominada matriz relacional difusa, cuyos elementos toman valores que se encuentran en el intervalo [0,1].

Atributos difusos:

Los atributos difusos se clasifican en tres tipos:

- datos precisos
- datos imprecisos sobre un conjunto de referencia ordenado. Estos atributos admiten tanto datos clásicos como difusos, en forma de distribuciones de posibilidad. Por ejemplo, la edad puede tener las etiquetas niño, joven, adulto, referenciadas sobre un conjunto entre 0 y 100.
- datos imprecisos sobre un conjunto de referencia normalizado pero no ordenado. Estos atributos son definidos sobre un dominio subyacente no ordenado, por ejemplo, el atributo “color del pelo” puede tener las etiquetas rubio, pelirrojo y castaño.

Cuantificadores difusos:

Los cuantificadores difusos permiten expresar cantidades o proporciones difusas para dar una idea aproximada del número de elementos de un subconjunto (o que cumplen cierta condición) o de la proporción de ese número en relación con el total de elementos posibles. Los cuantificadores pueden ser absolutos o relativos:

- Cuantificadores absolutos: expresan cantidades sobre el número total de elementos de un determinado conjunto, diciendo si este número es “grande”, “muchísimos”, “aproximadamente entre 5 y 10”, etc.
- Cuantificadores relativos: expresan mediciones sobre el número total de elementos que cumplen cierta característica dependiendo del total de elementos posibles, por lo que la verdad del cuantificador depende de dos cantidades. Este tipo de cuantificadores se usa en expresiones como “la mayoría”, “la minoría”, “aproximadamente 40 años”.

Base de reglas:

Un sistema de inferencia basado en lógica difusa, o sistema difuso, está formado por un conjunto del reglas del tipo “si x es A entonces y es B”, donde x e y son las variables del sistema, y A y B son términos lingüísticos como ‘alto’, ‘bajo’, ‘medio’, ‘positivo’, ‘negativo’, etc.

Mecanismo de inferencia:

Las técnicas de razonamiento utilizadas en lógica difusa son una generalización de los mecanismos de inferencia empleados en la lógica bi-valuada tradicional, pero en un sistema difuso varias reglas pueden estar activas simultáneamente con diferentes grados de activación. Cuando varía una de sus entradas el sistema evalúa todas las reglas para inferir una conclusión o salida. La regla composicional de inferencia proporciona un mecanismo para evaluar el resultado de una regla difusa. Un sistema de inferencia difuso contendrá un conjunto de reglas de descripción lingüística. En el caso más general los antecedentes y consecuentes de estas reglas incluirán proposiciones difusas compuestas, es decir, combinarán múltiples entradas y salidas. Este tipo de sistema se denomina sistema MIMO ("multiple input multiple output").

No obstante, un sistema MIMO siempre puede ser considerado como un conjunto de sistemas con entradas múltiples y una única salida, o sistemas MISO ("multiple input single output").

Dentro de estas técnicas, se encuentran, por ejemplo, el mecanismo de inferencia Min-Max (o método de Mamdani) o el método Producto-Suma.

Métodos de defuzzyficación:

En los mecanismos de inferencia anteriormente citados, el resultado de la inferencia es un conjunto difuso. Para que esta información pueda utilizarse en determinadas aplicaciones, como las de control, es preciso obtener un valor concreto representativo de dicho conjunto.

El proceso de defuzzificación se expresa mediante el operador defuzzificador F^{-1}que transforma la función de pertenencia representativa de un conjunto difuso μ(y) en un elemento concreto del universo de discurso.

Algunos de los métodos de defuzzificación existentes son el Centro de Gravedad (CoG), el Centro de Sumas (CoS), el Primer Máximo (FoM), el Último Máximo (LoM), la Media de Máximos (MoM), la Media Difusa (FM), la Media Difusa Ponderada (WFM), el Método de Calidad (QM), el Método "Level Grading" (LGM) o el Método de Yager (YM).

2.4. Teorías y modelos emocionales.

Estudios realizados en áreas como la psicología y la sociología han demostrado que las emociones cumplen un importante papel dentro de la inteligencia humana. Por otro lado, las investigaciones que se realizan en el campo de los agentes inteligentes han producido gran cantidad de agentes emocionales. Esto ha llevado a un nuevo planteamiento en el campo de la inteligencia artificial, incluyendo las emociones como una necesidad a cubrir en la mayoría de sistemas inteligentes, que responde a una pregunta que Minsky había planteado; "la cuestión no es si las máquinas inteligentes pueden tener emociones, sino si las máquinas pueden ser inteligentes sin emociones"[8]

Esto ha producido que en los últimos años se haya visto una expansión significativa en la investigación sobre los modelos computacionales de los procesos emocionales humanos, uniendo áreas tan diferentes como la psicología y la informática, que han generado modelos que compiten y se complementan entre sí.

En la siguiente figura se muestra la evolución histórica de algunos modelos más significativos[9]:

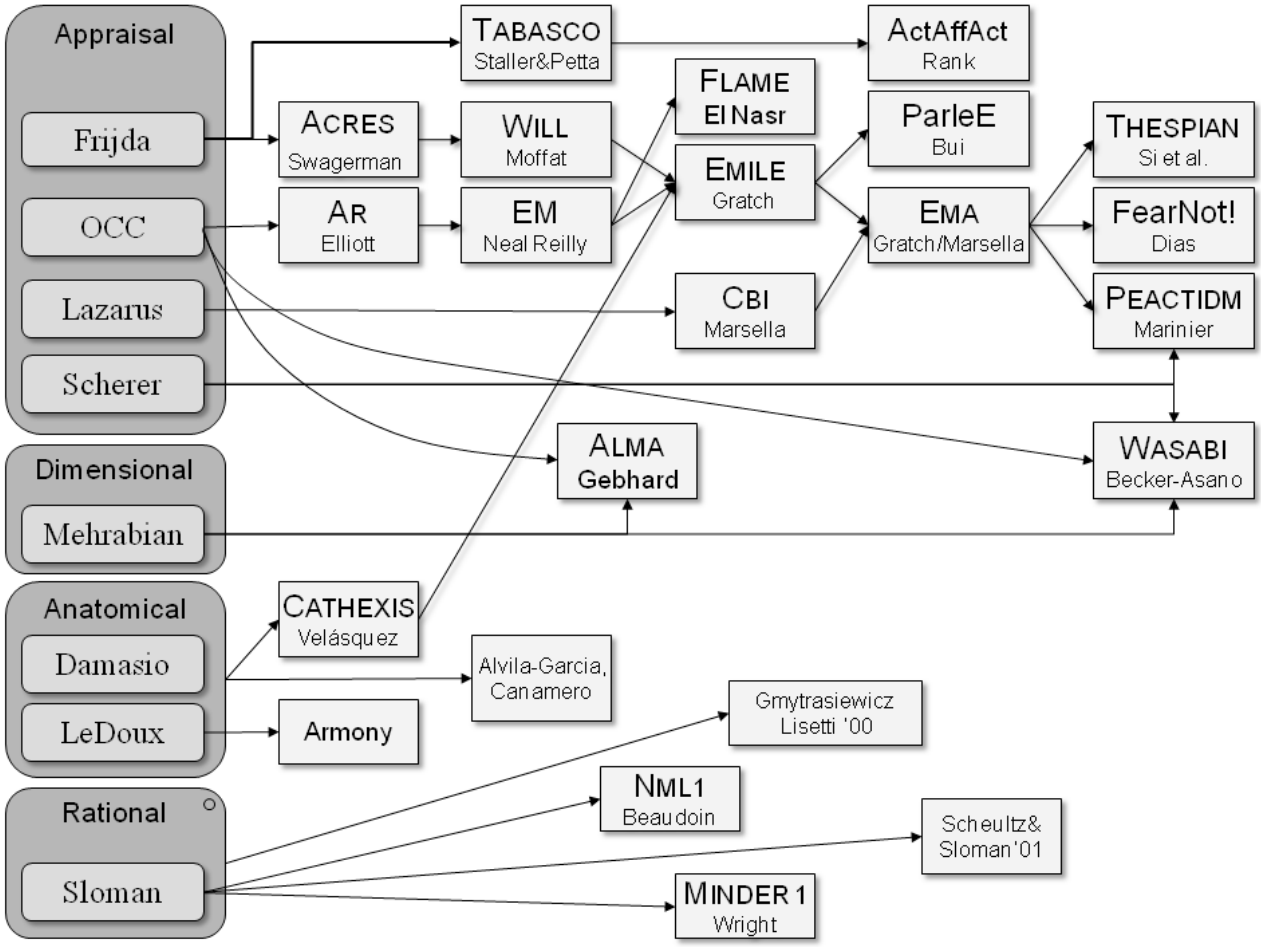


Ilustración 9: Cronología de los modelos Computacionales para representar emociones

Los modelos que figuran en esta ilustración se basan en diferentes teorías:

- Teoría de la apreciación (Appraisal Theory) : la mayoría de los modelos se basan en esta teoría, que enfatiza y explica la conexión entre la emoción y la cognición. En esta teoría, la emoción surge de unos patrones basados en juicios personales que relacionan los acontecimientos y las creencias, deseos e intenciones de un individuo. La mayoría de las aplicaciones de los modelos basados en esta teoría se utilizan para desarrollar personajes interactivos en tiempo real que puedan simular emociones para hacerlos más realistas y que sean capaces de entender e inducir en los usuarios comportamientos sociales positivos.

- Teoría dimensional (Dimensional Theory): esta teoría defiende que las emociones no se pueden conceptualizar de manera independiente, sino como puntos en un espacio continuo (normalmente tridimensional). En lugar de utilizar conceptos como alegría, miedo o ira, se hace hincapié en términos como estado de ánimo, en el que intervienen con diferente intensidad varios componentes por ejemplo placer-excitación-dominio (modelo tridimensional PAD Mehrabian and Russell 1974).

- Teoría anatómica (Anatomical Theory): Es un intento de reconstruir los enlaces neuronales y los procesos de las reacciones reflejas. Los modelos inspirados en esta teoría, separa las emociones en dos tipos, las emociones reflejas y las racionales y se centra las primeras.
- Teoría racional (Rational Theory): Esta teoría considera las emociones como un conjunto de procesos y limitaciones que responden a un determinado comportamiento de adaptación.
- Teoría comunicativa (Communicative Theory): Estas teorías afirman que los procesos emocionales son un mecanismo para comunicar a otros individuos del propio estado mental y así facilitar la coordinación social, y para pedir cambios en el comportamiento de los demás.

Como ya se ha comentado, la mayoría de los modelos emocionales se basan en la teoría de a apreciación. La siguiente ilustración muestra los componentes de este modelo computacional.

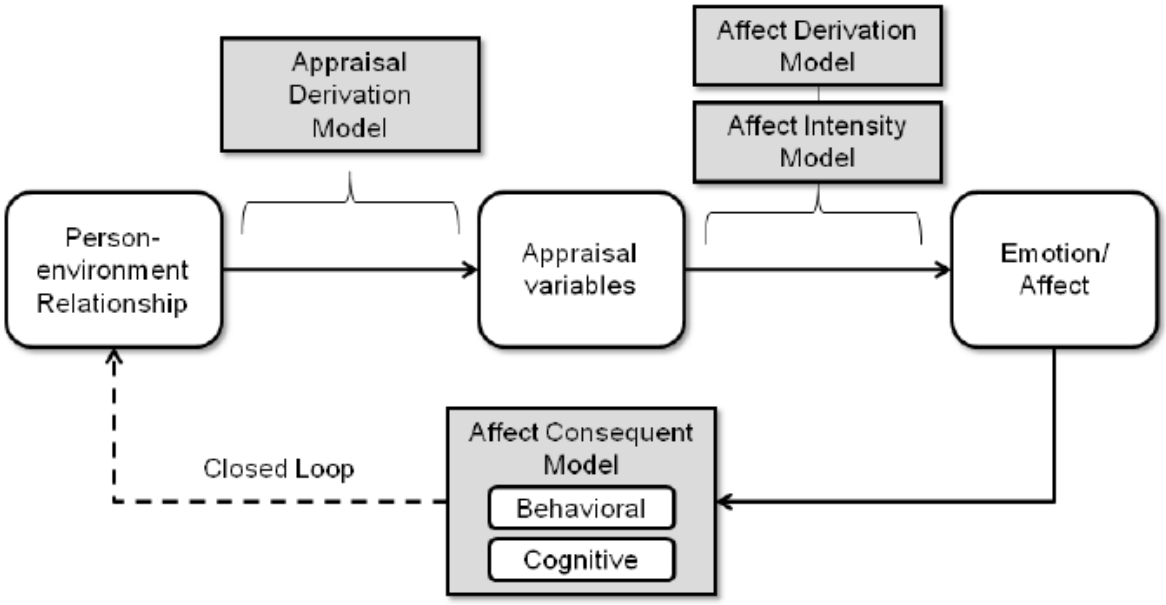


Ilustración 10: Componentes del modelo computacional basado en la teoría de la apreciación

Como se muestra en el modelo, la información tiene un carácter cíclico:

- La información del entorno es apreciada y representada mediante variables.
- Estas conducen a una respuesta afectiva de una intensidad determinada.
- Que produce una consecuencia en la conducta y la conciencia
- Y estas consecuencia alteran el entorno.

La **relación persona-entorno** se refiere a un tipo de representación que debe permitir a un agente relacionar los acontecimientos externos con sus creencias, deseos e intenciones.

El **modelo derivado de apreciación** transforma la representación de la relación persona-entorno en un conjunto de variables de apreciación. Esta transformación se puede hacer, por ejemplo, utilizando una serie

de reglas de inferencia independientes del dominio que derivan las características sintácticas de la relación persona-entorno en las variables de apreciación.

Las **variables de apreciación** son un conjunto de juicios específicos que el agente puede utilizar para producir diferente respuestas emocionales.

El **modelo derivado del afecto** especifica como va a reaccionar una persona emocionalmente basándose en un patrón de evaluación.

El **modelo de intensidad del afecto** especifica la intensidad de la respuesta emocional que resulta de una evaluación específica. A menudo el cálculo de la intensidad del afecto se implementa como parte del modelo derivado del afecto.

Afecto/emoción representa el estado actual emocional del agente. Esto podría estar representado por una etiqueta de emoción discreta, un conjunto de etiquetas de emociones discretas, un espacio dimensional continuo o incluso una combinación de estos factores.

Modelo consecuente del afecto produce un cambio conductual o cognitivo en el agente. La emoción puede dirigirse a exterior (a su entorno) o al interior dando forma a los pensamientos de una persona.

3. ESTUDIO DE LOS ACTUALES ENTORNOS VIRTUALES Y LAS HERRAMIENTAS QUE LOS SOPORTAN.

3.1. SecondLife

Second Life (abreviado SL) es un Metaverso altamente sofisticado, lanzado el 23 de junio de 2003, desarrollado por Linden Lab, y es accesible gratuitamente en internet. Sus usuarios conocidos como "residentes", pueden acceder a SL mediante el uso de uno de los múltiples programas de interfaz llamados viewers (visores), lo cual les permite interactuar entre ellos mediante un avatar. Los residentes pueden así explorar el mundo virtual, interactuar con otros residentes, establecer relaciones sociales, participar en diversas actividades tanto individuales como en grupo y crear y comercializar propiedad virtual y servicios en ellos.

Los avatares en SL son entidades de alta resolución no necesariamente humanos tridimensionales personalizables, lo que le da a los usuarios la capacidad de convertirse en el personaje que deseen y "gozar" de una segunda vida. Los avatares pueden ser diseñados para simular la apariencia real de sus usuarios en la vida real, ya que la calidad de estos lo permite. Sin embargo, el diseño está limitado por las herramientas que SL ofrece.

SL posibilita la creación de objetos para intercambiar a través de un mercado abierto que tiene como moneda local el Linden dólar ($L). Para ello el mismo programa incluye una herramienta de creación en 3D basada en simples figuras geométricas (prims o primitivas). A los objetos así creados se les puede añadir funcionalidades mediante un lenguaje de programación propio de SL el Linden Scripting Language (LSL). También da la posibilidad de crear objetos más complejos e importarlos a SL.

La programación de SL es abierta y libre. El código permite a los usuarios modificar casi cualquier aspecto de sus apariencia y comportamiento en el mundo virtual, e incluso la construcción de elementos tridimensionales.

Los avatares en SL son meras marionetas, solo generan la ilusión de que poseen algún tipo de mentalidad, no obstante la creación de Agentes Virtuales Autónomos es posible según se demuestra en el siguiente ejemplo que se publicó en el portal de la ciencia y la tecnología en Español, el 28 de marzo de 2008.

El artículo[10] publicado, se refiere a un proyecto de investigación, respaldado por IBM y otros patrocinadores, que lleva a cabo el grupo del investigador Selmer Bringsjord, director del Departamento de Ciencias Cognitivas del Renssealaer,.

Ilustración 11: Los AVAs en Second Life

Se trata de un avatar, desarrollado como un agente virtual autónomo que representa a un niño de 4 años (Eddie) dotado del razonamiento adecuado a su edad. Para probar los poderes de

razonamiento de Eddie, el grupo creó una versión de demostración en Second Life que sometió su teoría a una prueba de creencia incorrecta.

Este proyecto se ha conseguido utilizando un superordenador y uniendo inteligencia artificial y técnicas de modelado cognitivo computacional, la teoría de la mente. Esta teoría se refiere a los principios y las técnicas que los humanos despliegan para comprender, predecir y manipular el comportamiento de otros humanos. El grupo de investigación de Bringsjord trabaja sobre esta teoría, que incluirá definiciones de todos los conceptos centrales dentro de dicha teoría de la mente, incluyendo la mentira, el mal o la traición.

SL es altamente interactivo (incluye interacción visual y auditiva con dispositivos de entrada como micrófonos y webcam) y participativo, y está dotado de gran realismo, esto ha propiciado que empresas de diferentes sectores del mundo real tengan presencia en Second Life, dónde también ofrecen sus productos y servicios.

Un ejemplo es el de CEF que abrió en 2007 una sede en Second Life, convirtiéndose de esta manera en la primera escuela española con presencia en el universo SL.

El atractivo de SL en el área de la educación es que el entorno fomenta un alto grado de participación del alumno al ofrecer la posibilidad de realizar actividades en tiempo real, con sus compañeros y profesores, desde cualquier parte del mundo y con una alta sensación de realidad.

Esta sede virtual cuenta con una superficie de 1.024 metros cuadrados y está dividida en dos plantas.

Ilustración 12: Sede virtual de CEF en Second Life

Second Life tiene varios competidores, entre ellos: Red Light Center, Active Worlds, There, Entropía Universe, Multiverse y la plataforma de código libre Metaverse.

3.2. Otros mundos virtuales

3.2.1. Active Worlds.

Active Worlds es una comunidad virtual dónde sus usuarios pueden chatear y construir entornos de realidad virtual en 3D en millones de kilómetros cuadrados de territorio virtual. Los usuarios pueden hacer compras online en un centro comercial de realidad virtual y chatear con empleados del comercio pero también pueden jugar a juegos interactivos en 2D y 3D.

Los avatares son en 3D, pero no tiene moneda local y el chat es de texto.

3.2.2. There.

There es un mundo virtual 3D en linea que se lanzó en 2003. Contiene varios lugares exóticos para explorar. Todos los avatares son humanos y se pueden personalizar, además los usuarios pueden crear sus propios artículos, como ropa, coches, edificios, muebles, etc.

Al igual que Second Life tiene su propia moneda local, chat de texto y VoIP.

Se cerró en marzo del 2010, aunque en mayo de este año se anunció que volvería a abrir.

3.2.3. The Sims Social

Software de relaciones interpersonales de humanos virtuales, en el cual el usuario tiene que tratar de mantener el equilibrio emocional de, por lo menos, el personaje que controla, a través de la cotidianidad de los días y de sus relaciones con otros humanos virtuales.

3.2.4. Metaverse.

Metaverse es un proyecto de código abierto para desarrollar una plataforma compartida multiusuario de mundos virtuales en linea del estilo Second Live. Fue fundado en 2004 por Hugh Perkins y Jorge Lima.

El proyecto tenía como objetivo producir un motor de código abierto, flexible, escalable y altamente configurable para la creación de mundos en 3D con la capacidad de conectar múltiples mundos virtuales existentes basados en estándares metaverso y de código abierto.

La aplicación está constituida por un servidor y un componente cliente. El servidor se puede instalar, configurar y administrar por cualquier persona para crear su propio metaverso. El cliente se puede comunicar con cualquier servidor de metaverso compatible.

A partir del 2008 el proyecto dejó de funcionar.

3.2.5. Solipsis.

Solipsis, es un sistema de código abierto para mundos virtuales multiusuario compartido. Fue diseñado por joaquin Keller y Gwendal Simon y lanzado en 2007 en código abierto. Su objetivo es proporcionar la infraestructura para un metaverso público.

La diferencia con otros metarversos es que solipsis utiliza una arquitectura descentralizada (peer-to-peer) para evitar el colapso que supone la utilización de servidores privados.

El mundo inicialmente está vacío, y son los propios usuarios quienes deben ir completándolo.

3.3. Lenguajes para el modelado de mundos virtuales.

3.3.1. VRML.

VRML es un Lenguaje para Modelado de Realidad Virtual (Virtual Reality Modeling Language)[11], sirve para crear mundos en tres dimensiones a los que poder acceder con un navegador mediante un pluggin capaz de visualizar archivos VRML, que es un formato de archivo normalizado que tiene como objetivo la representación de escenas u objetos interactivos y tridimensionales.

El lenguaje VRML posibilita la descripción de una escena compuesta por objetos 3D a partir de prototipos basados en formas geométricas básicas o de estructuras en las que se especifican los vértices y las aristas de cada polígono tridimensional y el color de su superficie.

El Consorcio Web3D fue creado para desarrollar este formato. Su primera especificación fue publicada en 1995 y la versión actual es la VRML 97 que es la base en la se ha desarrollado X3D (Extensible 3D).

Características de VRML

Es un lenguaje que permite el modelado de estructuras tridimensionales con la cuales el usuario puede interactuar y obtener información. Entre otras cosas VRML permite:

- Habilitar múltiples sensores en el mundo virtual.
- La animación para que los objetos adquieran movimiento dentro del mundo
- Reproducir sonidos y películas dentro de los mundos creados.
- Que lo usuarios interactúen con con el mundo creado.
- Controlar y mejorar los mundos mediante scripts en otros lenguajes de programación.

El modelado en VRML lo proporcionan los nodos con los que se construyen objetos 3D, se crean luces, se aplican texturas a los objetos, se detecta la proximidad de un objeto a otro, etc. Para animar un mundo virtual se pueden conectar unos nodos a otros, de manera que puedan intercambiar información por medio de eventos. Estos eventos pueden estar producidos de forma automática o como respuesta de un objeto a otros eventos que haya recibido. Por ejemplo, al pulsar sobre un objeto en pantalla con el ratón se puede hacer que el objeto desaparezca.

VRML está diseñado para usarse sobre internet, intranets y sistemas locales. Los mundos VRML pueden transmitirse e inter-relacionarse a través del WWW y visualizarse mediante algún browser VRML que se conecta con el browser WWW a través de un API.

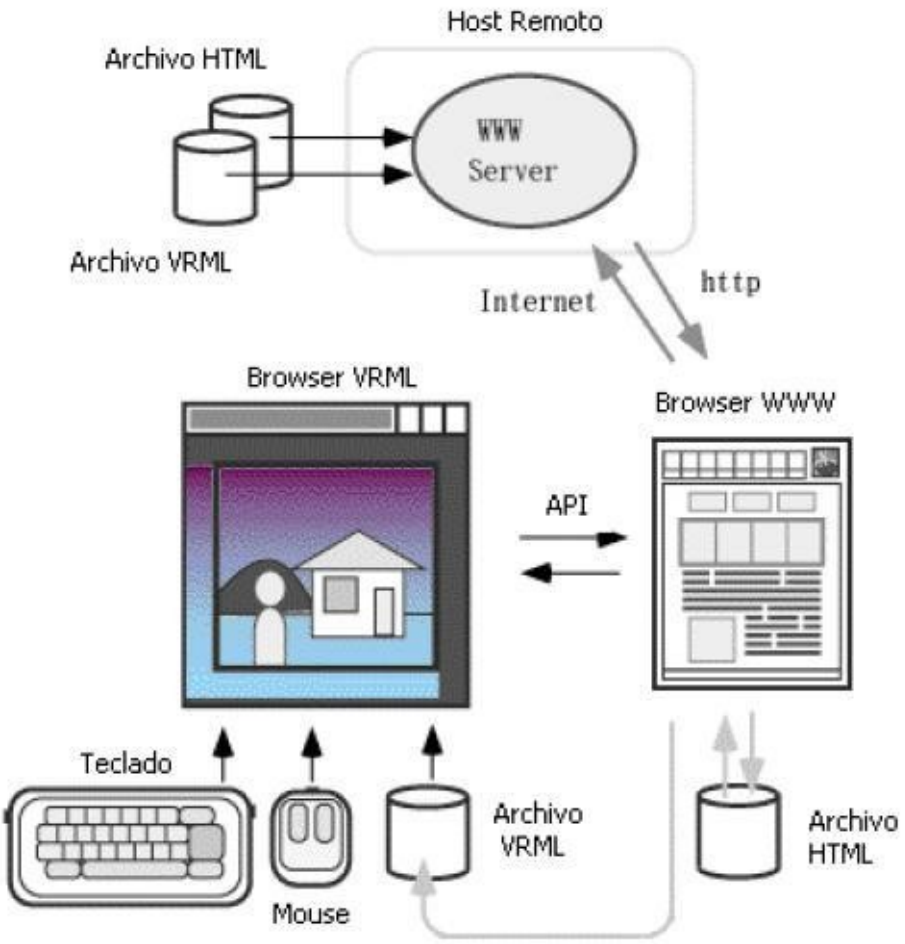


Ilustración 13: Arquitectura del sistema de comunicación entre Mundo VRML.

¿Como crear mundos con VRML?

Para crear mundos virtuales con VRML, sólo se necesita un editor de texto y para luego visualizarlo es necesario un navegador VRML o un pluggin (visualizador VRML) instalado en cualquier navegador. Un ejemplo de visualizador para Netscape Navigator y Internet Explorer bajo plataformas win32 es Cortona VRML Client.

Los documentos VRML son ficheros de texto que proveen al navegador de las instrucciones necesarias para modelar y renderizar las figuras 3D contenidas en el mundo virtual.

Hay aplicaciones específicas para la creación de mundos virtuales que están basadas en VRML y además muchos editores gráficos 3D pueden convertir las creaciones con ellos realizadas al formato VRML.

Los ficheros VRML tienen la extensión .wrl (punto world) para indicar que contiene un mundo virtual VRML.

Estructura básica de un fichero VRML.

- La cabecera VRML que indica la versión, tiene carácter de comentario a iniciarse con el símbolo #, pero al ser la primera línea es una excepción y es leída por el navegador. Ejemplo *#VRML v2.0 UTF8*

- Comentarios. Se inician con el símbolo #, pueden situarse en cualquier posición de una linea y terminan con el retorno de carro.

- Nodos. Son los elementos básicos del lenguaje. Cada nodo define una característica de la escena (un objeto 3D, una textura, un sensor, …), y a su vez puede ser parte de otro nodo, es decir que pueden anidarse entre si, creando una jerárquica de nodos que suele llamarse Grafo de la Escena. Cada nodo tendrá una serie de atributos que lo caracterizan.

  Cada nodo consta de un nombre que identifica el tipo de nodo de que se trata y, entre llaves, otros nodos o atributos con sus valores.

  Un nodo básico es el WorldInfo, que contiene la información general sobre el mundo.

  En el siguiente tabla extraída de http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/ aparecen todos los nodos posibles en VRML.

6.1 Introduction	6.20 FontStyle	6.39 ScalarInterpolator
6.2 Anchor	6.21 Group	6.40 Script
6.3 Appearance	6.22 ImageTexture	6.41 Shape
6.4 AudioClip	6.23 IndexedFaceSet	6.42 Sound
6.5 Background	6.24 IndexedLineSet	6.43 Sphere
6.6 Billboard	6.25 Inline	6.44 SphereSensor
6.7 Box	6.26 LOD	6.45 SpotLight
6.8 Collision	6.27 Material	6.46 Switch
6.9 Color	6.28 MovieTexture	6.47 Text
6.10 ColorInterpolator	6.29 NavigationInfo	6.48 TextureCoordinate
6.11 Cone	6.30 Normal	6.49 TextureTransform
6.12 Coordinate	6.31 NormalInterpolator	6.50 TimeSensor
6.13 CoordinateInterpolator	6.32 OrientationInterpolator	6.51 TouchSensor
6.14 Cylinder	6.33 PixelTexture	6.52 Transform
6.15 CylinderSensor	6.34 PlaneSensor	6.53 Viewpoint
6.16 DirectionalLight	6.35 PointLight	6.54 VisibilitySensor
6.17 ElevationGrid	6.36 PointSet	6.55 WorldInfo
6.18 Extrusion	6.37 PositionInterpolator	
6.19 Fog	6.38 ProximitySensor	

Ilustración 14: Nodos en VRML

Construcción de objetos tridimensionales.

Existen varias técnicas para construir objetos sólidos tridimensionales en VRML:

- Mediante Primitivas: son objetos predefinidos a los que únicamente hay que especificar sus parámetros, estos objetos son; cubo, cilindro, cono, esfera y texto.

- Superficies aproximadas por polígonos: una superficie compleja se puede aproximar mediante un número suficiente grande de polígonos.

- Mapas de altura: definiendo una malla de puntos en el plano horizontal a los cuales se les especificará una altura.

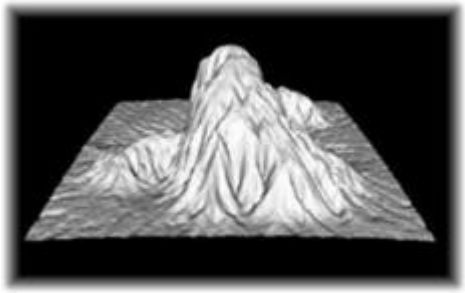


Ilustración 15: Superficies complejas mediante mapas de altura en VRML

Otras características.

- **Posicionamiento en el espacio:** Todo fichero VRML consta de un sistema de coordenadas tanto para los objetos definidos en él como para los que sean incluidos desde una fuente externa. Todos los objectos se crean en el centro del sistema de referencia del entorno virtual. Para situar un objeto en otra posición distinta o para cambiar su orientación en la escena se utiliza el nodo *Transform*.
- **Apariencia:** Mediante el nodo Appearance se pueden determinar algunas características de la apariencia de un objeto determinado o de parte de él: material, textura, suavizado de aristas, etc.
- **Luces:** VRML permite situar múltiples fuentes luminosas en una escena. Estas fuentes pueden simular el sol, la luz eléctrica, la luz de una llama, etc. Las fuentes de luz se pueden situar en un punto determinado del espacio y hacer que iluminen en una determinada dirección y con una determinada intensidad y color.

  Las fuentes luminosas en VRML son de tres tipos: Fuentes direccionales (DirectionalLight), fuentes puntuales (PointLight) y fuentes focales (SpotLight). Para cada una de ellas se puede indicar, si está encendida o apagada en un instante determinado, su intensidad, su efecto ambiental y su color. Para las fuentes puntuales y las focales se puede definir un alcance máximo y un factor de atenuación en función de la distancia.

Elementos para la Animación.

En VRML es posible crear objetos en movimiento o con partes móviles, para hacerlo de sebe indicar cómo y cuándo el objeto debe efectuar su movimiento, lo que requiere un modelo interno de ejecución que gobierne cómo cambian las cosas y en qué orden lo hacen. Esto se hace conectando los nodos entre sí, proporcionando rutas.

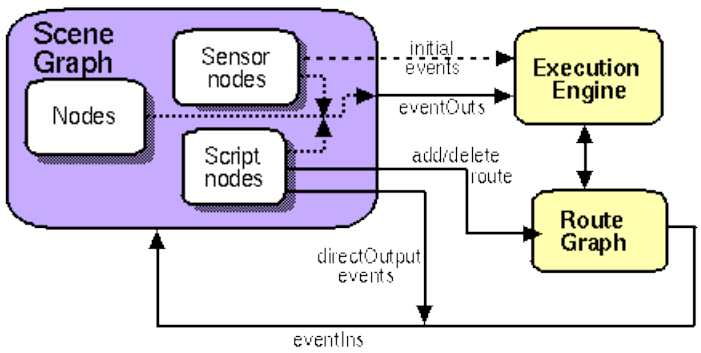


Ilustración 16: Modelo de ejecución en VRML

- **Rutas**: Son conexiones entre nodos, no tienen una representación visible pero integran los elementos que constituyen el mundo VRML. A través de ellas se pueden enviar y recibir eventos.
- **Eventos**: Los eventos son mensajes que ligan los elementos de la escena y todo lo que se mueve o interactua en VRML se debe a los eventos.

  Un evento tiene dos partes: El mensaje (un valor o dato) y el time stamp (corresponde al momento preciso en que se produjo el evento.

  Casi todos los nodos poseen eventos de entrada y de salida (eventIns y eventOuts) que es la manera en que los nodos se comunican con el exterior. Los eventIns son receptores que escuchan los eventos del exterior y los procesan. Los eventOuts son transmisores que envían los eventos producidos por el nodo al exterior.

Los nodos que normalmente se utilizan para la interacción y la animación en VRML son los siguientes:

- **Sensores**: Existen dos tipos de sensores; sensores para el usuario y sensores medioambientales.
  - Los sensores para el usuario sirven para obtener entradas de datos del usuario. Estos sensores detectan pulsaciones de ratón, operaciones de arrastrar y soltar y otras operaciones similares. Algunos ejemplos son:
    - TouchSensor: Detecta la interacción del ratón sobre un objeto.
    - SphereSensor: Permite rotar un objeto.
    - PlaneSensor: Permite arrastrar el objeto a través de un plano bidimensional.
  - Lo sensores medioambientales captan eventos dentro de la escena, como el paso del tiempo, la posición del usuario y otras.
    - TimeSensor: Es un cronómetro abstracto. Puede usarse para generar eventos regulares.
    - VisibilitySensor: Para evitar que estén activadas siempre todas las animaciones del mundo. Las animaciones estarán activadas cuando estén dentro del campo de visión.
    - ProximitySensor: Es muy similar a un VisibilitySensor
    - Collision: Se usa para detectar colisiones entre el usuario y objetos, evitando que los pueda atravesar, dando así realismo a la escena.
- **Interpoladores:** Los nodos interpoladores sirven para generar animaciones. Un interpolador recoge la señal del TimeSensor y realiza una interpolación lineal entre un juego de valores llamados keyValues, que son los valores a modificar.
- **Scripts**: El comportamiento de algunos objetos puede ser demasiado complejo para ser representado mediante los nodos anteriores. Para ello se dispone del nodo Script, en el que se puede definir cualquier objeto con los campos, eventos de entrada y eventos de salida necesarios.

El nodo Script también permite la creación de elementos que actúen como interfaz del usuario con el mundo virtual, como barras de desplazamiento, cuadros de textos, etc.

A partir del 2005 se empezó a promover el uso de X3D, que es un formato de archivo 3D con habilitación XML para comunicar en tiempo real datos de información 3D a través de todas las aplicaciones.

Tanto VRML como X3D son estándares, pero no son compatibles, es decir un navegador que sea puramente x3D no será capaz de leer archivos VRML, aunque la mayoría de navegadores soportan ambos estándares.

3.3.2. X3D.

X3D[12] es un estándar abierto XML, un formato de archivo 3D que permite la creación y transmisión de datos 3D entre distintas aplicaciones y, especialmente, aplicaciones en red.

Características:

- X3D está integrado en XML: esto representa un paso fundamental a la hora de conseguir una correcta integración en:
  - Servicios Web.
  - Redes distribuidas.
  - Sistemas multiplataforma y transferencia de archivos y datos entre aplicaciones.
- X3D es Modular: tiene componentes
- X3D es Extensible: permite añadir componentes para ampliar las funcionalidades.
- X3D es Perfilado: se pueden escoger distintos grupos de extensiones según las necesidades específicas de la aplicación.
- X3D es Compatible con VRML

3.3.3. Java3D.

Java 3D es un API de gráficos 3D desarrollada por Sun como una extensión del JDK del lenguaje de programación Java. Es una colección de clases que tienen como objetivo principal facilitar la creación y representación de escenas tridimensionales en el ordenador, así como la animación e interacción con las mismas.

Java 3D utiliza el concepto de Grafo de una Escena para la representación de escenas tridimensionales. En este grafo se describen la geometría de los objetos y sus propiedades (colores, texturas, movimientos, etc.), su ubicación dentro de la escena, orientación, fuentes de luz, lugar en donde está situado el observador, etc.

Para construir un mundo virtual en Java 3D, debemos crear y manipular objetos geométricos tridimensionales que se encuentran en un universo virtual que después es renderizado. El renderizado se hace en paralelo gracias al aprovechamiento de los Threads de Java. Los programas pueden escribirse

para ejecutarse como aplicaciones o como applets en navegadores.

Escenario Gráfico:

Es donde se crea un universo virtual en Java 3D. Se trata de una estructura de datos de tipo árbol en donde hay un nodo raíz. La relación más común entre nodos es padre-hijo. El otro tipo de relación es por referencia que asocia un objeto con un nodo del escenario gráfico.

Los escenarios gráficos tienen solamente un VirtualUniverse, que a su vez tiene un objeto Locale que proporciona referencias a un punto en el escenario virtual y sirve de raíz para varios sub-gráficos. Los objetos BranchGroup son la raiz de los sub-gráficos y tienen dos categorías: Por un lado está la rama de contenido gráfico que especifica el contenido del universo virtual (apariencia, comportamiento, geometría, etc.) y por el otro está la rama de vista gráfica, que especifica los parámetros de visualización (posición y dirección).

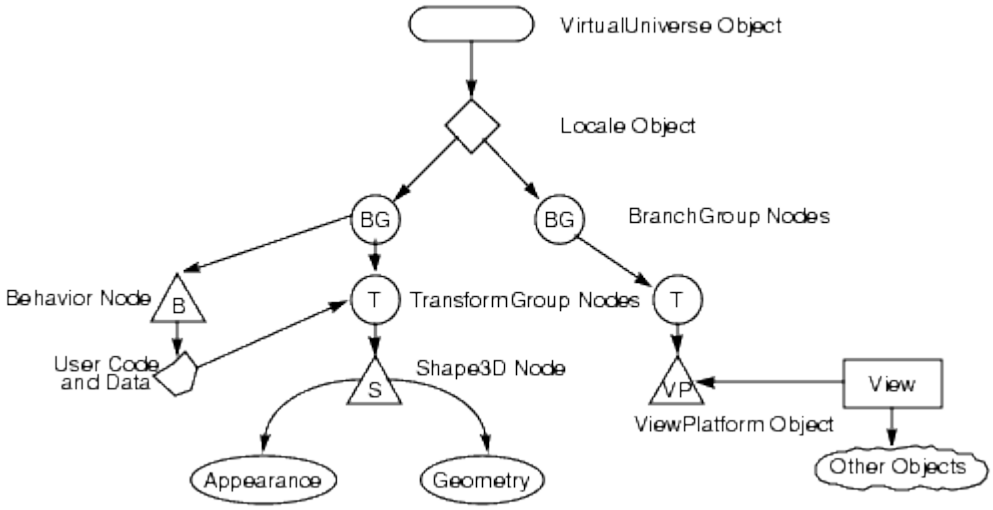


Ilustración 17: Escenario Gráfico en Java 3D

El comportamiento en Java3D

Los objetos de la clase Behavior son los encargados de la interacción y la animación. Esta es una subclase abstracta que proporciona mecanismos para incluir código que modifique el escenario gráfico, los gráficos y los sonidos del universo virtual. El estímulo que cambia el escenario gráfico puede ser una pulsación de tecla, un movimiento de ratón, la colisión de objetos o el paso del tiempo. También puede provocarse la adición de objetos al escenario gráfico, su eliminación, cambio de atributos de los objetos, reordenación, etc.

Java3D y VRML/X3D

Java 3D dispone de cargadores (de VRML y X3D que traducen ficheros de este formato en objetos apropiados en Java3D.

Ventajas de VRML/X3D sobre Java3D:

- La creación de escenarios simples en código Java3D es más engorroso que la creación de la

misma escena en VRML o X3D.

- La pérdida de velocidad y prestaciones de Java3D frente a los visores VRML/X3D desarrollados en C++ y empleando directamente Direct3D u OpenGL.

Ventajas de Java3D sobre VRML/X3D:

- E control de los distintos elementos presentes en el sistema es superior y más natural en Java3D. En VRML crear una interacción con el usuario es más complejo. Para ello, es necesario elegir entre el uso de programación interna dentro del propio código VRML/X3D, que es muy costosa y limitada, o de programación externa. Además está sujeto a la implementación de la especificación VRML/X3D que haya realizado el creador del visor VRML/X3D que se esté empleando.
- No necesita la instalación de un visualizador como en el caso de VRML y X3D.

Java3D como visor de archivos VRML/X3D:

Para ello solo es necesario emplear un loader (cargador) de archivos VRML/X3D desarrollado para Java3D, como Xj3D desarrollado por el Web3D Consortium bajo licencia GNU LGPL (Lesser General Public License). Las ventajas principales de emplear Java3D como visor de VRML/X3D son la capacidad de ejecución en distintas plataformas y el liberar al usuario final de la necesidad de instalar un plug-in específico para el navegador.

3.4. Visualizadores de mundos VRML/X3D.

Los browsers para mundos VRML/X3D son plug-ins que permiten visualizar objetos realizados con VRML desde cualquier navegador de internet.

3.4.1. Cortona 3D.

Se trata de un visor de 3D interactivo para visualizar mundos virtuales en la web. Es compatible con con diversas tecnologías para desarrollo 3D y con todos los formatos de VRML.

El programa dispone de las opciones habituales de los visores 3D para el web, como pueden ser la selección de las distintas vistas, movimiento en primera persona, estudio de objetos, etc.

Cortona está desarrollado por la empresa Parallel Graphics, es gratuito y se puede descargar en la web oficial de la empresa. También en la misma web se pueden encontrar ejemplos de funcionamientos de los mundos VRML.

Cortona 3D se instala con facilidad, y su uso está muy extendido. Sin embargo no dispone de las librerías VRML para el API de Java EAI, por lo que no soporta la conexión con programas externos escritos en Java, si soporta JavaScrip para programación dentro del fichero VRML aunque resulta algo limitada, al no permitir la totalidad de las funciones como lo hacen otros visualizadores.

3.4.2. Cosmo Player.

Fue el primer navegador que soportó VRML 2.0, con sensores, scripts y sonido. Está optimizado para su uso con páginas Web.

Tiene una interfaz muy sofisticada, y dispone del API Java EAI para VRML por lo que se puede desarrollar programas externos en Java o JavaScript.

3.4.3. FreeWRL.

Es un proyecto del sourceforge.net para visualizar mundos VRML/X3D tanto como aplicación de escritorio como a través de la web, que soporta las especificaciones VRML de la Web3D Consortium.

Está desarrollado para las plataformas linux, Mac y también se exportó a Windows, aunque esta última está en estado Beta, y no funciona correctamente.

La mayor ventaja es que es código abierto.

3.5. Herramientas para la creación de mundos virtuales.

La mayoría de las herramientas actuales son para la creación de videojuegos.

3.5.1. EasyVRML.

EasyVRML era un ambiente CASE, orientado a la generación de mundos virtuales con comportamiento complejo. Permite llevar a cabo la asignación de este tipo de comportamientos descritos en Java o C++ a componentes de mundos virtuales creado con VRML a través de un lenguaje visual basado en iconos. Actualmente en desuso por falta de actualización a entornos X3D.

3.5.2. Vivaty Studio.

Es una herramienta interactiva para la creación de mundos X3D/VRML que permite al usuario crear todos los aspectos de un mundo X3D con una interfaz gráfica de usuario. Actualmente está en estado Beta.

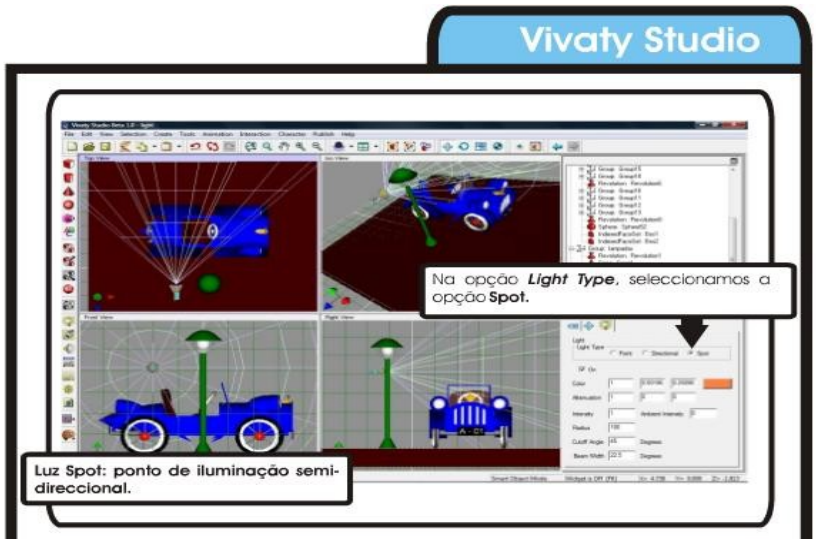


Ilustración 18: Herramienta para crear mundos VRML/X3D: Vivaty Studio

3.5.3. OpenSimulator.

Open Simulator es una plataforma para crear mundos virtuales 3D. Permite implementar un servidor que soporte un mundo virtual similar al de Second Life al que otra personas se pueden conectar a

través de internet.

Necesita un cliente para conectarse al mundo virtual que puede ser el cliente oficial de SecondLife.

OpenSim tiene versiones para Linux, Windows, Mac y FreeBSD..

3.5.4. Unity 3D.

Motor para la creación de videojuegos. Permite importar sonidos y personajes animados. También permite modificar los scripts generados por la misma herramienta para personalizar aspectos y comportamientos de la escena.

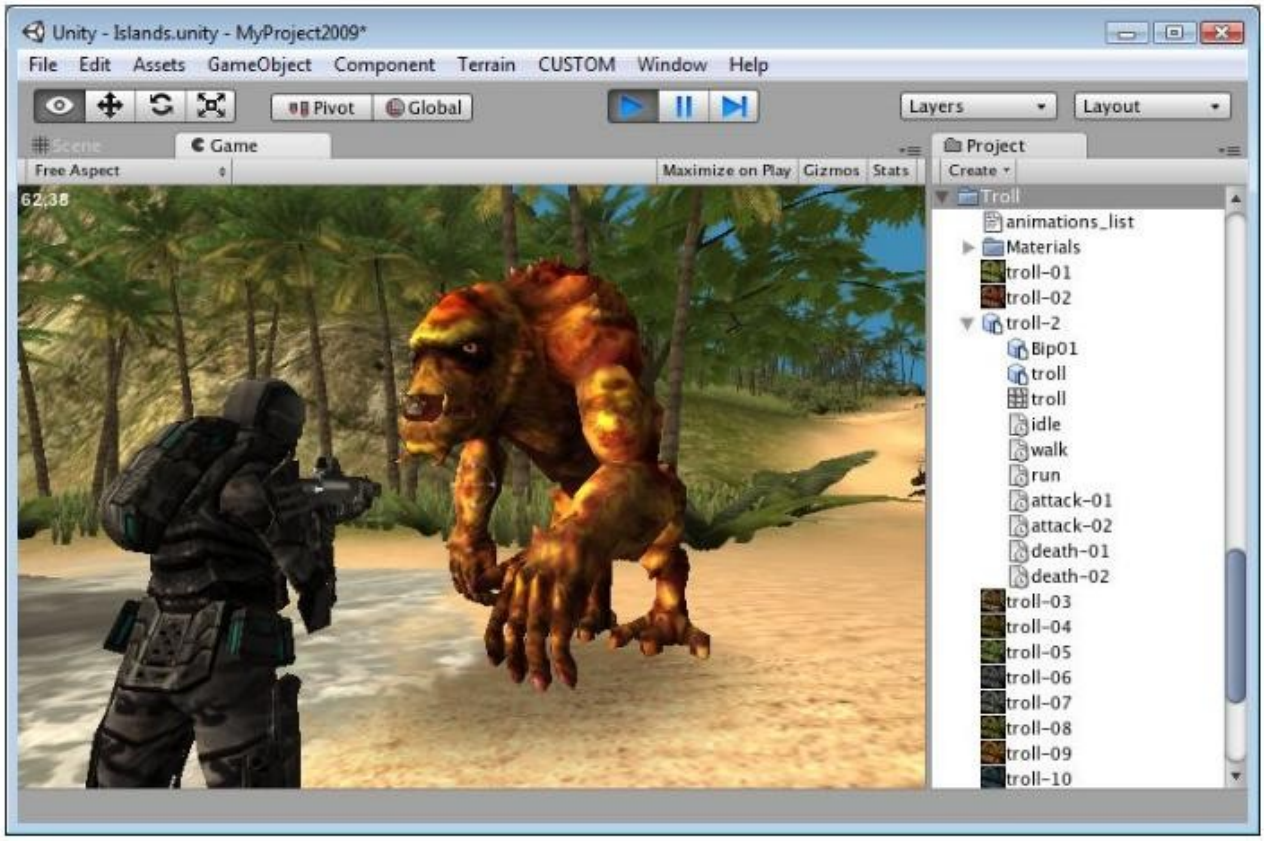


Ilustración 19: Motor para crear videojuegos: Unity 3D

3.5.5. 3D GameStudio.

3D GameStudio es un kit de desarrollo comercial de juegos de ordenador. Consta de un motor 3D, un motor 2D, un editor de niveles y modelos, un compilador de scripts y librerías de modelos, texturas, etc. Tiene un motor de iluminación que soporta sombras verdaderas y fuentes de luz en movimiento. El principal objetivo que esta aplicación persigue es que el desarrollador del juego no necesite ser un programador experimentado. Se reduce al máximo el esfuerzo de desarrollo a costa de perder flexibilidad en el diseño del juego. La última versión es del 2002.

4. ANÁLISIS DE LA POSIBILIDAD DE CREACIÓN DE ENTORNOS VIRTUALES INTELIGENTES EN LA ACTUALIDAD, CONTEMPLANDO EL FACTOR TEMPORAL.

El incremento de poder computacional permite que una parte de poder de procesamiento dentro de un entorno virtual pueda ser dedicado a la adición de inteligencia.

Los entornos virtuales inteligentes son aquellos en los que se pone énfasis en incrementar las capacidades de comportamiento del entorno y de los objetos y personajes que lo componen y su interacción con él, lo que se consigue mediante la incorporación de sistemas de Inteligencia Artificial que interactúen con el sistema gráfico.

La incorporación de personajes inteligentes en estos entornos, implica además una necesidad de representación del conocimiento.

4.1. El comportamiento en Entornos Virtuales.

El comportamiento se refiere a la modificación del estado de los objetos dentro de un escenario virtual, en respuesta a alguna acción del usuario o al cambio de otros objetos de la escena o, incluso a los cambios motivados de manera intrínseca por el mismo objeto que modifica su comportamiento con el transcurso del tiempo.

En un Entorno Virtual VRML/X3D se pueden diferenciar dos tipos de comportamientos:

- Comportamiento simple: Los cambios en el estado de los objetos dependen exclusivamente de eventos internos generados los nodos de la escena.
- Comportamiento complejo: Los cambios dependen de un programa escrito en un lenguaje ajeno a VRML/X3D.

Los nodos scripts de un entorno virtual modelado por VRML/X3D pueden proporcionar algunos comportamientos complejos pero tiene muchas limitaciones por lo que para comportamientos inteligentes será necesaria la utilización de otros lenguajes como java o C++, dos de los lenguajes orientados a objetos más extendidos.

Tanto java como C++ disponen de librerías para la programación de agentes de software inteligentes. Además los dos lenguajes de programación soportan que la localización física del código que define el comportamiento sea local o remota, es decir que se pueda ejecutar en la misma máquina donde está la escena VRML o en cualquier máquina en la red.

Sin embargo, C++ no es soportado de manera directa por VRML para describir comportamiento, para comunicarse sino que se deben definir los mecanismos de interfaz entre ellos.

Por otro lado, Java, además de ofrecer un acceso casi transparente a la red y una comunicación con VRML completamente funcional, es un lenguaje independiente de la plataforma a utilizar. Por lo que será la mejor

opción para construir entornos virtuales inteligente. El acceso a la red se realiza mediante el protocolo www: Un fichero html contendrá un applet Java y embebido el mundo VRML.

```
<embed src="box.wrl" height="300" width="300">
  <applet code="box.class" mayscript height="2" width="2">
  </applet>
```

Ilustración 20: Código html para mostrar un mundo VRML y ejecutar un applet de Java.

4.2. Especificación del API de Java EAI para VRML.

Para lograr la integración de Java con VRML, es necesario utilizar la External Authoring Interface (EAI), la cual permite la comunicación y el paso de parámetros entre mundos virtuales VRML y los programas escritos en Java. Cada visor VRML dispone de un paquete de clases VRML específico.

En Cosmo Player el paquete de clases se llama npcosmop21.jar y deberá estar situado en el directorio de Java \jre\lib\ext\ para poder compilar el programa Java que lo utiliza. Y es compatible con la especificación VRML/EAI. A continuación se describe la jerarquía de clases de esta especificación:

```
vrml.external
  |
  +- vrml.external.IBrowser
  |       +- vrml.external.Browser
  |       +- vrml.external.CBrowser
  |
  +- vrml.external.Node
```

Para conseguir la comunicación de una aplicación Java con un mundo VRML, lo primero que es necesario es obtener una instancia del Browser, mediante la clase vrml.external.Browser se consigue una instancia de la escena existente, esta clase se puede instanciar de dos maneras diferentes:

```
public Browser(Applet pApplet);
public Browser(Applet pApplet, String frameName, int index);
```

Ilustración 21: Código para instanciar una escena VRML en Java

Las clases field.EventIn se encargan de obtener datos de los campos de los diferentes nodos que componen la escena.

```
+- vrml.external.field
|       +- vrml.external.field.EventIn
|       |       +- vrml.external.field.EventInMFColor
|       |       +- vrml.external.field.EventInMFFloat
|       |       +- vrml.external.field.EventInMFInt32
|       |       +- vrml.external.field.EventInMFNode
|       |       +- vrml.external.field.EventInMFRotation
|       |       +- vrml.external.field.EventInMFString
|       |       +- vrml.external.field.EventInMFVec2f
|       |       +- vrml.external.field.EventInMFVec3f
|       |       +- vrml.external.field.EventInSFBool
|       |       +- vrml.external.field.EventInSFColor
|       |       +- vrml.external.field.EventInSFFloat
|       |       +- vrml.external.field.EventInSFImage
|       |       +- vrml.external.field.EventInSFInt32
|       |       +- vrml.external.field.EventInSFNode
|       |       +- vrml.external.field.EventInSFRotation
|       |       +- vrml.external.field.EventInSFString
|       |       +- vrml.external.field.EventInSFTime
|       |       +- vrml.external.field.EventInSFVec2f
|       |       +- vrml.external.field.EventInSFVec3f
|       |
```

Por el contrario las clases field.EventIn se encargan de modificar los campos de los diferentes nodos que componen la escena.

```
|       |
|       +- vrml.external.field.EventOut
|       |       +- vrml.external.field.EventOutMField
|       |       |       +- vrml.external.field.EventOutMFColor
|       |       |       +- vrml.external.field.EventOutMFFloat
|       |       |       +- vrml.external.field.EventOutMFInt32
|       |       |       +- vrml.external.field.EventOutMFNode
|       |       |       +- vrml.external.field.EventOutMFRotation
|       |       |       +- vrml.external.field.EventOutMFString
|       |       |       +- vrml.external.field.EventOutMFVec2f
|       |       |       +- vrml.external.field.EventOutMFVec3f
|       |       |
|       |       +- vrml.external.field.EventOutSFBool
|       |       +- vrml.external.field.EventOutSFColor
|       |       +- vrml.external.field.EventOutSFFloat
|       |       +- vrml.external.field.EventOutSFImage
|       |       +- vrml.external.field.EventOutSFInt32
|       |       +- vrml.external.field.EventOutSFNode
|       |       +- vrml.external.field.EventOutSFRotation
|       |       +- vrml.external.field.EventOutSFString
|       |       +- vrml.external.field.EventOutSFTime
|       |       +- vrml.external.field.EventOutSFVec2f
|       |       +- vrml.external.field.EventOutSFVec3f
|       |
|       +- vrml.external.field.EventOutObserver
|       +- vrml.external.field.FieldTypes
|
```

Por último las clases exception definen las posibles excepciones.

```
|
+- vrml.external.exception
        +- vrml.external.exception.InvalidEventInException
        +- vrml.external.exception.InvalidEventOutException
        +- vrml.external.exception.InvalidNodeException
        +- vrml.external.exception.InvalidVrmlException
```

4.3. Agentes de software inteligentes.

Los agentes inteligentes son programas que actúan de forma autónoma como resultado de la observación de su entorno. Mediante los sistemas basados en agentes inteligentes podemos abordar las diferentes áreas de un problema, relacionándolas entre si. De esta forma, la representación, la búsqueda y resolución de problemas, el aprendizaje automático y el razonamiento pueden ser abordadas con este paradigma.

Los agentes funcionan de forma independiente, pero entre ellos deben poder comunicarse con tal de abordar un problema intercambiando información. Un agente estará en constante búsqueda de los datos necesarios para llevar a cabo su función gracias a su capacidad de contemplar su entorno y responder ante éste.

Los agentes inteligentes pueden ser reactivos o deliberativos según como reaccionen a los estímulos externos, pueden hacerlo solo basándose en reglas básicas o con capacidad de razonamiento e iniciativa.

Los agentes actúan creando un modelo que define que acciones se desean realizar para obtener un fin y como obtenerlos. Para ello intervienen la visión del mundo del agente (las creencias), qué posibilidades tiene según esas creencias (deseos) y qué objetivos va a conseguir (intenciones).

Dado el posible estado cambiante del entorno de un agente en aplicaciones interactivas, el agente debe revisar su visión del mundo, para poner al día sus objetivos y las acciones que debe realizar. Esto puede suponer un problema en función del volumen de información con la que deba trabajar ya que actualizar los objetivos/acciones puede ser muy costoso, así que debe buscarse un equilibrio en la frecuencia de actualización de este proceso.

En Java es posible la implementación de estos agentes con la utilización de las librerías de JADE (Java Agent Development Framework.

JADE es también una plataforma para ejecutar agentes que permite la ejecución completamente asíncrona y la comunicación entre agentes en la misma o diferentes plataformas.

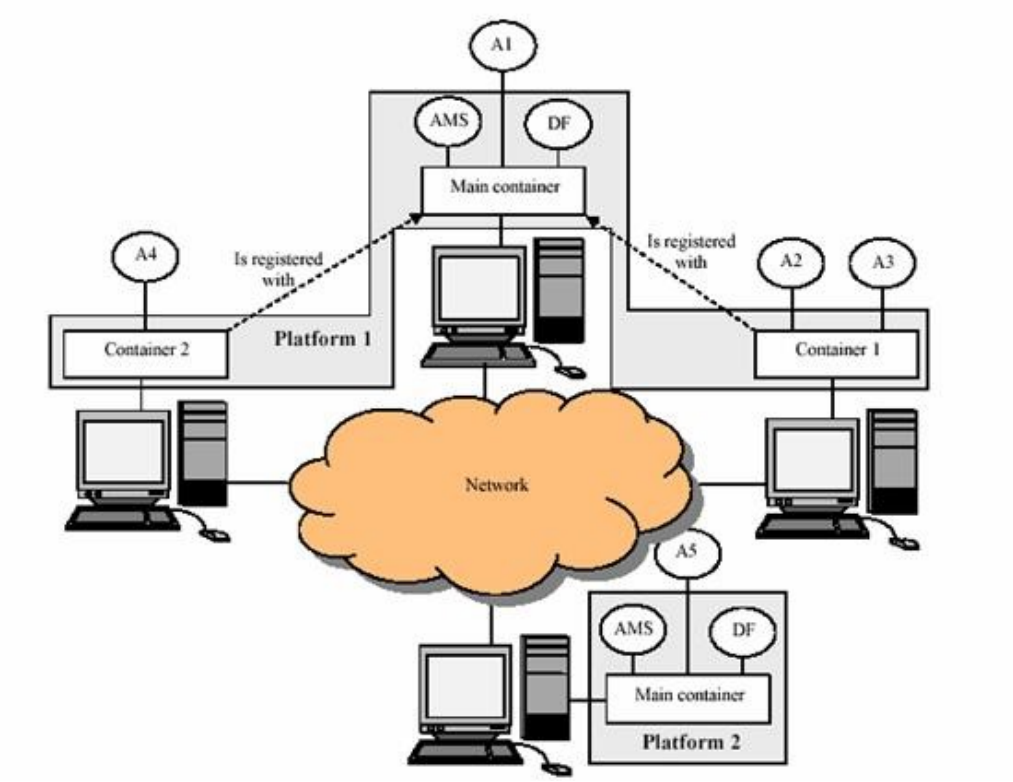


Ilustración 22: Arquitectura JADE

JADE facilita las comunicaciones, pero no estructura de razonamiento interno, se utiliza más para agentes reactivos, pero existe una extensión JADEX que facilita la implementación del modelo deliberativo de comportamiento.

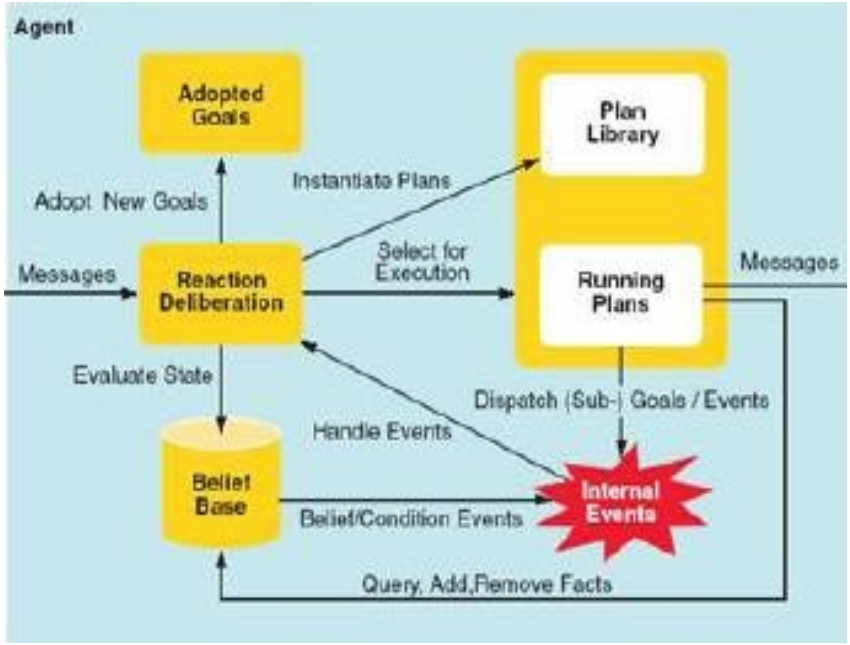


Ilustración 23: Arquitectura JADEX

Por otro lado JADE se puede combinar con JESS o con DROOLS (o Jboss Rules), que son sistemas de gestión de reglas de negocio, para dotar a los agentes de inteligencia, o con la librería jFuzzyLogic de Java.

4.4. Sistema de inferencia basado en lógica difusa.

Para poder proporcionar comportamiento inteligente a los agentes autónomos en un mundo virtual será necesario la utilización de herramientas que permitan evaluar los estímulos que el agente recibe de forma continua en el tiempo y modelar su comportamiento dentro del su entorno. El sistema que implementa esta capacidad es la inferencia y la herramientas para su desarrollo lo hacen sobre la base teórica de la lógica difusa.

4.4.1. Fuzzy Control Langage FCL.

Es un lenguaje simple para definir un sistema de inferencia difuso (FIS) que se utiliza en las clases de Java de la librería jFuzzyLogic. FCL es un estándar incluido en el IEC 61131.

Este sistema de inferencia se almacena en un fichero con la extensión .fcl que puede ser cargado y utilizado por un programa en Java.

La estructura de un fichero fcl, consta de uno o más bloques de función, en cada bloque se definen 1 o más variables de entrada y de una o más variables de salida y de un bloque de reglas, A continuación se describe los términos básicos que se utilizan en FCL:

- FUNCTION_BLOCK: Una función se define en un bloque, en un fichero puede haber más de un bloque, es decir se pueden definir más de una función.
- VAR_INPUT y VAR_OUTPUT: Se definen las variables de entrada y salida que deben ser variables reales,
- FUZZIFY: Cada una de las variables de entrada se transforman en uno o más términos (TERM) con los que se define una función de pertenencia.
- DEFUZZIFY: De la misma manera que con la variables de entrada en este bloque se hace con las variables de salida.
- RULEBLOCK: Para definir las reglas.
- METHOD: Para definir el método de defuzzificar.

Keyword	Explanation
COG	Centre of Gravity (Note 1)
COGS	Centre of Gravity for Singletons
COA	Centre of Area (Notes 2 and 3)
LM	Left Most Maximum
RM	Right Most Maximum

La siguiente ilustración muestra la estructura de funcionamiento de un sistema FIS[13]:

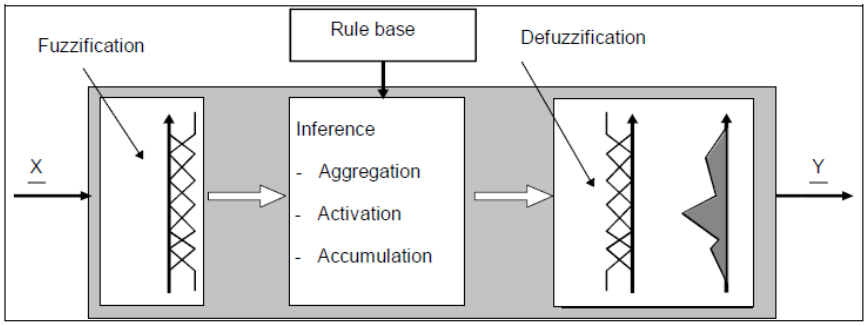


Ilustración 24: Estructura y elementos funcionales de un control difuso en FCL.

4.4.2. JFuzzyLogic.

La librería JfuzzyLogic es un paquete escrito en Java bajo licencia LGPL que implementa el lenguaje de control difuso FCL de acuerdo con la especificación IEC 61131 part. 7.

El paquete está formado por las siguientes clases:

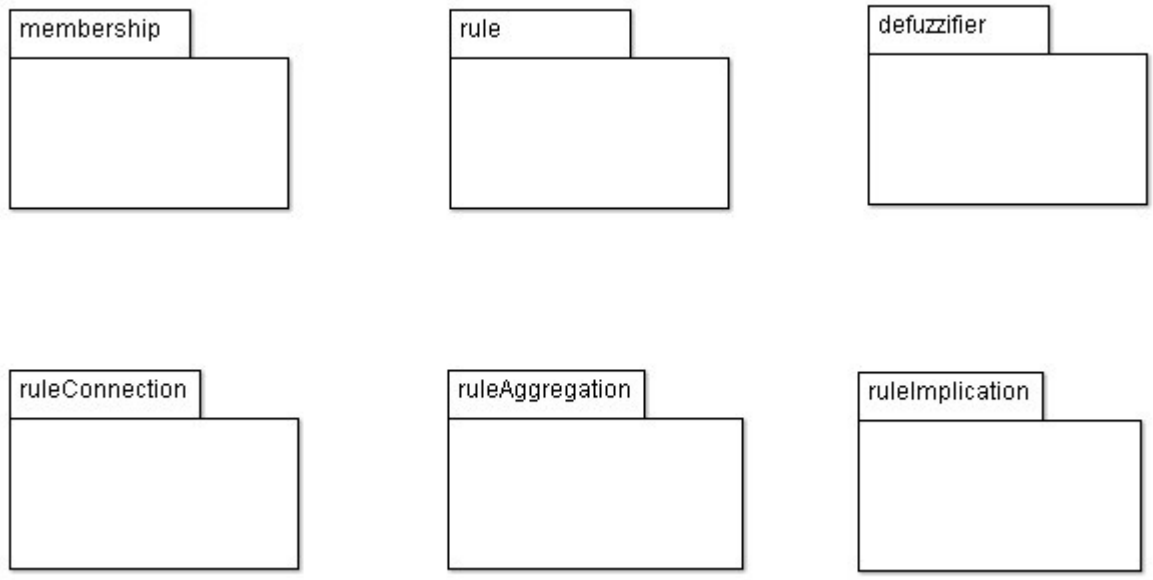


Ilustración 25: JFuzzyLogic Package

El funcionamiento es muy sencillo. Una vez definido el fichero FCL, se carga el sistema de inferencia al programa en Java utilizando el método ***load*** de la clase **FIS[14].**

Una vez cargado el sistema de inferencia se puede evaluar con los datos que se pasan como parámetros al método setVariable de la clase FIS.

El resultado de la evaluación se obtiene mediante el método getVariable de la clase FIS.

4.4.3. FLAME - Fuzzy Logic Adaptive Model of Emotions

En un capítulo anterior se introdujo como investigaciones en diferentes campos de estudio habían desarrollado una serie de modelos computacionales para representar y simular las emociones. Entre los modelos citados se encuentra FLAME[15] un modelo que utiliza la lógica difusa.

FLAME se basa en el modelo de la apreciación y usa las reglas de la lógica difusa para evaluar el impacto de los eventos en los objetivos de la intensidad emocional. También incluye varios algoritmos de inducción para conocer las expectativas, recompensas, los patrones de las acciones del usuario, las asociaciones objeto-emoción, etc. Estos algoritmos pueden permitir a un agente inteligente, implementado utilizando FLAME, adaptarse dinámicamente a los usuarios y a su entorno.

FLAME también se basa en el modelo OCC para producir etiquetas discretas de emociones.

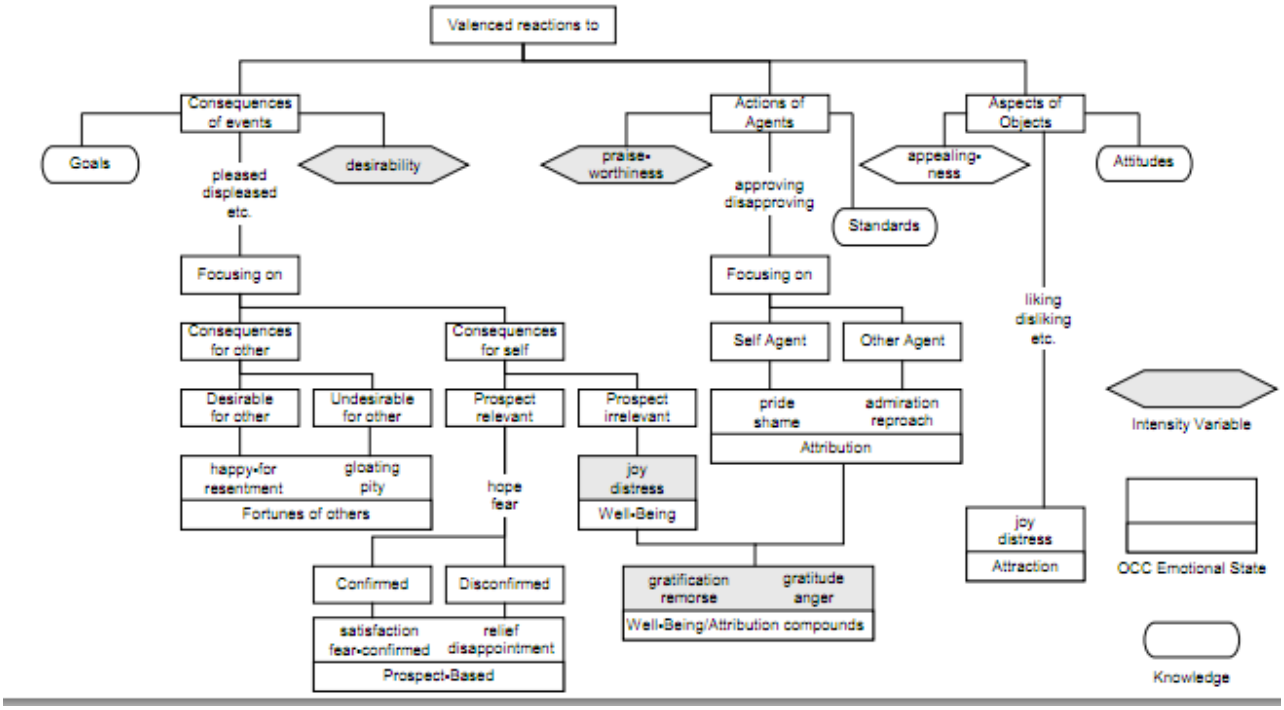


Ilustración 26: Modelo OCC de las emociones

El modelo OCC se ha auto establecido como el modelo estándar para la síntesis de las emociones.

La siguiente ilustración representa la arquitectura abstracta de un agente emocional en el modelo FLAME:

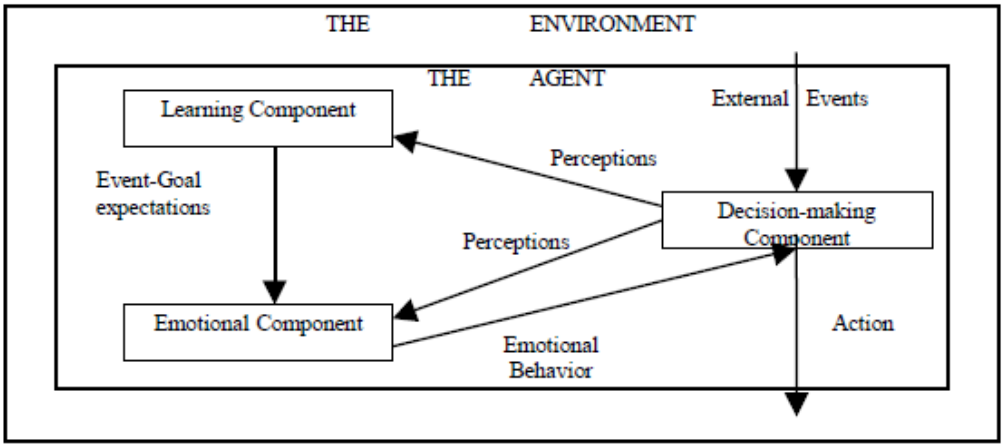


Ilustración 27: Arquitectura de un agente emocional en el modelo FLAME

El modelo consta de tres componentes: un componente emocional, un componente de aprendizaje y un componente de toma de decisiones.

El agente percibe los acontecimientos externos de su entorno y estas percepciones pasan al componente emocional y al componente de aprendizaje. El componente procesa las percepciones y además utiliza algunos resultados del componente de aprendizaje, que son las expectativas y las metas que producen un comportamiento emocional. Los resultados los devuelve al componente de toma de decisiones para elegir una acción. La acción a tomar tiene en cuenta el estado y la conducta emocional del agente desencadenando así una acción.

En la siguiente figura se representan los componentes del proceso emocional:

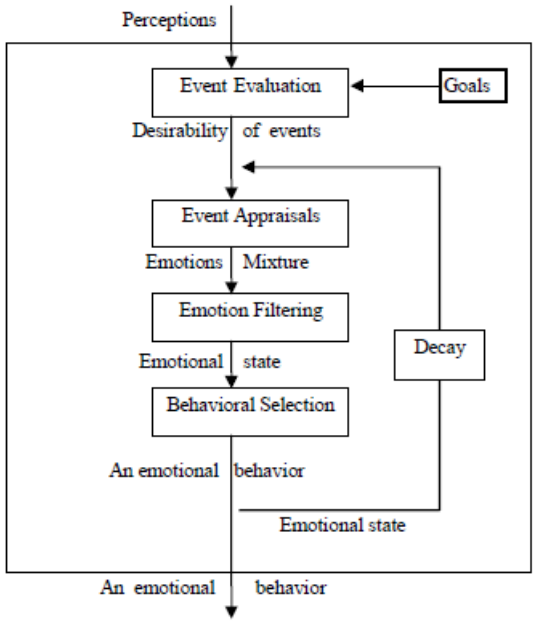


Ilustración 28: Componentes del proceso emocional en el modelo FLAME

El proceso emocional está dividido en varios subprocesos, representados en la figura como rectángulos, y tal como se muestra la información pasa de un subproceso a otro.

En primer lugar se evalúan las percepciones provenientes del entorno. El proceso de evaluación consta de dos pasos secuenciales. El modelo de la experiencia determina que objetivos se ven afectados por el evento y el grado de impacto que el evento tiene en estos objetivos, luego, las reglas de asignación calculan el nivel de conveniencia del evento de acuerdo con el impacto calculado por el primer paso y la importancia de los objetivos en cuestión. Se suelen utilizar reglas difusas para determinar la conveniencia de un evento de acuerdos con esto dos criterios.

- **Evaluación de eventos:** Mediante reglas difusas se deduce la conveniencia de los acontecimientos y su impacto en los objetivos y la importancia de estos objetivos. El impacto de un evento en un objetivo se describe con cinco conjuntos difusos.

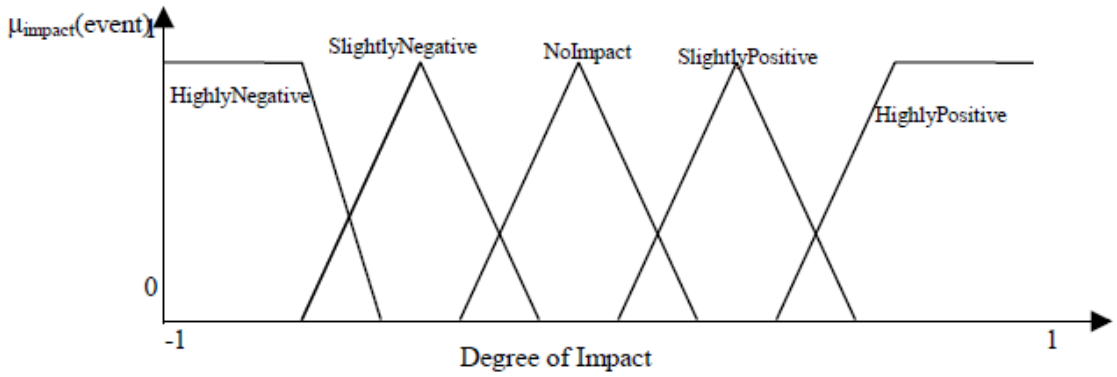


Ilustración 29: Representación de los conjuntos difusos para evaluar el impacto de un objetivo en el proceso emocional del modelo FLAME

La importancia de un objetivo se establece dinámicamente de acuerdo con la apreciación del agente respecto a una situación particular. La medida de la importancia de un objetivo está representado por tres subconjuntos difusos tal como se muestra en la siguiente figura:

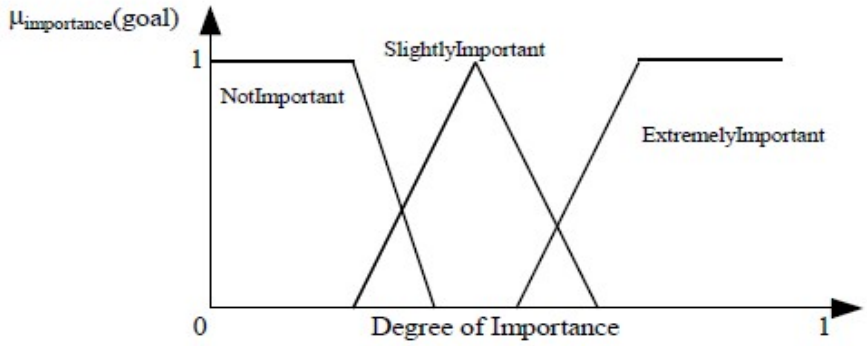


Ilustración 30: Representación de los conjuntos difusos para evaluar la importancia de un objetivo en el componete emocional del modelo FLAME

Por último la conveniencia de la medida de los eventos puede ser medido con los siguientes subconjuntos difusos:

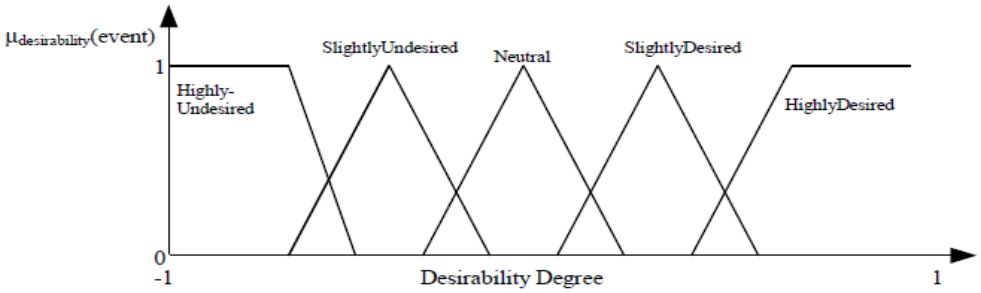


Ilustración 31: Representación de los conjuntos difusos para evaluar la conveniencia de un evento en el proceso emocional del modelo FLAME

Para la inferencia se puede utilizar diferentes modelo por ejemplo Mamdani con defuzificación centroide y para calcular los grados de composición de cada regla Max-Min.

El proceso de defuzzificación devolverá un número que luego se utiliza como una medida de la conveniencia del evento de entrada.

- **Apreciación de eventos:** Una vez que la conveniencia de un acontecimiento está determinada, se evalúan las reglas para determinar el estado emocional, que también tiene en cuenta las expectativas. Los valores esperados son derivados del modelo de aprendizaje que no se va a tratar en este trabajo. Las relaciones entre las emociones, expectativas, y la conveniencia de un evento se basa en las definiciones presentadas por Ortony (1988), y se presentan en la siguiente tabla:

Emotion	Rule
Joy	occurrence of a desirable event
Sad	occurrence of an undesirable event
Disappointment	occurrence of a disconfirmed desirable event
Relief	occurrence of a disconfirmed undesirable event
Hope	occurrence of an unconfirmed desirable event
Fear	occurrence of an unconfirmed undesirable event
Pride	action done by the agent and is approved by standards
Shame	action done by the agent and is disapproved by standards
Reproach	action is done by the other and is not approved by the agents standards
Admiration	action done by the other and is approved by the agents standards
Anger	Complex emotion; sad + reproach
Gratitude	Complex emotion; joy + admiration
Gratification	Complex emotion; joy + pride
Remorse	Complex emotion; sad + shame

Ilustración 32: Reglas para la emociones definidas por Ortony

En la tabla están modeladas catorce emociones. Emociones como el amor y el odio hacia otros se miden por las acciones de los otros y como ayuda al agente para lograr sus objetivos.

La cuantificación de la intensidad de las emociones provocadas por estas reglas se puede calcular las formulas presentadas en la siguiente tabla:

Emotion	Formula for Intensity
Joy	$Joy = (1.7 \times expectation^{0.5}) + (-0.7 \times desirability)$
Sadness	$Sad = (2 \times expectation^{2}) - desirability$
Disappointment	$Disappointment = Hope \times desirability$
Relief	$Relief = Fear \times desirability$
Hope	$Hope = (1.7 \times expectation^{0.5}) + (-0.7 \times desirability)$
Fear	$Fear = (2 \times expectation^{2}) - desirability$

Ilustración 33: Fórmulas para calcular la intensidad de una emoción en el proceso emocional del modelo FLAME

La tabla muestra el método para calcular la intensidad para diversas emociones teniendo en cuenta el valor de una expectativa y la medida de la conveniencia de un evento.

- **Filtrado de emociones:** Las emociones suelen aparecer mezcladas. Por ejemplo, el sentimiento de tristeza se mezcla a menudo con la vergüenza, la ira o el miedo. Las emociones, a veces son inhibidas o mejoradas por los estados de motivación. En general el filtrado emocional puede ser de dominio-dependencia y la influencia de otros factores más complicados como la personalidad. En el modelo FLAME, el método de filtrado de las emociones depende de los estados motivacionales. Los estados motivacionales tienden a interrumpir el desarrollo del proceso cognitivo para satisfacer una meta más alta. El proceso emocional siempre busca la mejor emoción para expresar en diferentes situaciones. El estado de ánimo también puede ayudar a filtrar esa mezcla de emociones desarrolladas. Las emociones negativas y positivas se tienden a influir entre sí sólo cuando el estado de ánimo está en el límite entre dos estados.

- **Decaimiento:** Al final de cada ciclo las emociones del agente reducen su intensidad. Este proceso es importante para que el modelo emocional sea realista. Normalmente las emociones no desaparecen una vez su causa ha desaparecido, sino que se van desintegrando a través del tiempo. El modelo FLAME utiliza diferentes constantes para la descomposición de las emociones dependiendo si son positivas o negativas.

- **Selección del comportamiento:** La lógica difusa se utiliza una vez más para determinar el comportamiento basado en un conjunto de emociones. El comportamiento depende del estado emocional del agente y la situación o el sucesos que ocurrió.

5. DISEÑO DE UN ENTORNO VIRTUAL INTELIGENTE.

En este apartado se describe el análisis y diseño del problema objeto de este PFC, que supone la realización de un prototipo simple de un entorno virtual con unas variables de salida que serán recogidas por un sistema difuso para modelar el comportamiento de un agente inteligente inmerso en dicho entorno virtual. Es decir que por un lado hay que diseñar gráficamente el entorno virtual con VRML y por otro lado hacer el análisis y diseño del sistema difuso y su implementación en Java.

5.1. Análisis y diseño del sistema inteligente.

En el apartado anterior se ha planteado la utilización del modelo de representación de las emociones y la utilización de la lógica difusa para inferir un determinado comportamiento y en el apartado de conceptos básicos de esta memoria del PFC se ha explicado con detalle como se puede modelar un sistema de inferencia mediante lógica difusa.

A continuación se pondrá en práctica este modelado para resolver el problema planteado en este proyecto.

5.1.1. Arquitectura del sistema inteligente.

Basada en la arquitectura del modelo FLAME de la ilustración se propone la siguiente arquitectura para nuestro agente inteligente:

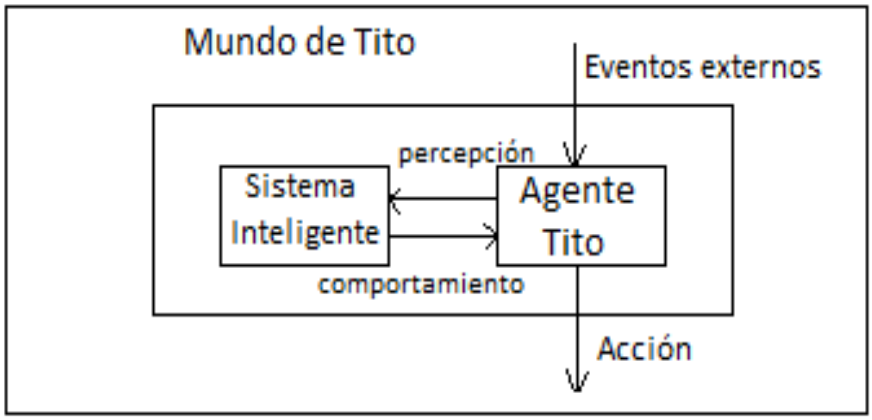


Ilustración 34: Arquitectura del prototipo

Se ha suprimido el componente de aprendizaje, puesto que el problema a resolver no contempla esta variable.

El sistema inteligente de la ilustración anterior, que en el modelo FLAME, se denomina componente emocional quedará definido de la siguiente manera:

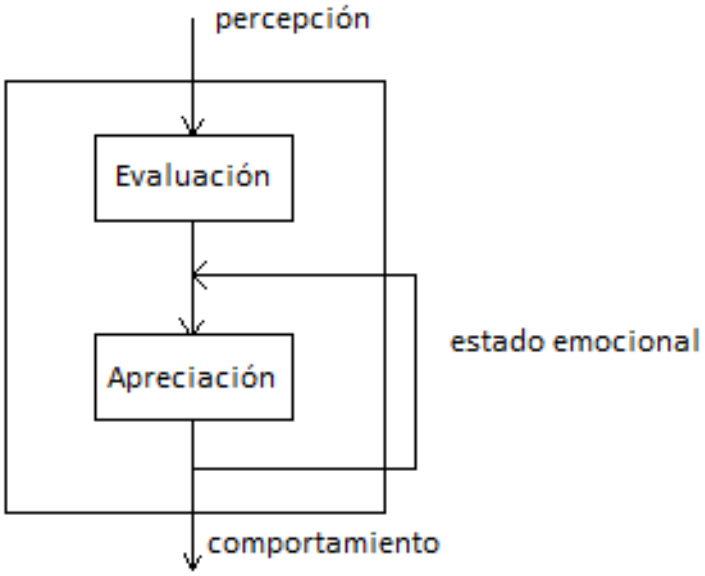


Ilustración 35: Sistema inteligente

5.1.2. Motor de inferencia. Sistema difuso.

Como ya se ha indicado anteriormente, los sistemas difusos son sistemas basados en reglas en los que éstas se definen en términos de conjuntos difusos.

Un sistema basado en reglas sigue la estructura siguiente:

si <premisa> entonces <conclusión>

En los sistemas difusos tanto las premisas como las conclusiones corresponden a conceptos vagos.

En el sistema a resolver son premisas: muchas personas, pocas personas, algunas personas, feliz, triste, etc. Y conclusiones: alegre, triste, muy alegre, muy triste, indiferente... Que son valores de las variables lingüísticas utilizadas en el sistema.

Formalmente, una variable lingüística viene definida por una tupla de la forma <X, LX, UX, SX>, donde los elementos que aparecen corresponden a:

- X: el nombre de la variable lingüística.
- L_X: los valores lingüísticos que puede tomar la variable X.
- U_X: es el universo de discurso sobre el cual toma valores la variable lingüística.
- S_X: es la función semántica que da significado (interpreta) a cada uno de los términos lingüísticos. Así, a cada valor de LX se le asigna un conjunto difuso sobre UX.

En el problema a resolver se suponen dos variables lingüísticas de entrada; población y estado inicial del

agente. Y una variable de salida; estado anímico final del agente.

La población es el número de personajes que hay en escena en un momento dado t, aparte del agente inteligente que se ha llamado Tito. El numero de personajes en escena en cada momento variará en función de un algoritmo interno del entorno que se describirá en un apartado posterior de esta memoria.

La variable estado inicial (estadoi), es el estado anímico de Tito en un periodo t-1 anterior al periodo que se está considerando.

Para obtener los valores de la variable estadoi se tendrá en cuenta los valores Red y Green del color que tiene Tito en el t-1. Estos valores están comprendidos entre 0 y 1 para cada uno, se transformarán a un solo valor que será el resultado de la siguiente fórmula:

$$estadoi = \begin{cases} 1 - valorRed & si \;\; valorRed > valorGreen \\ 1 & si \;\; valorRed = valorGreen \\ valorGreen + 1 & si \;\; valorRed < valorGreen \end{cases}$$

El rojo absoluto representado por los valores RGB (1, 0 , 0) representa el estado de ánimo muy triste que estará representado por el valor 0 de la variable estadoi y el verde absoluto (0, 1, 0) el estado de ánimo muy alegre (estadoi será 2). El color amarillo se consigue con los valores RGB (1, 1, 0) por lo que el valor de entrada de la variable estadoi será 0.

Definición de la variable lingüística de entrada población:

<Población, $L_{población}$ = {nadie, pocos, algunos, muchos, demasiados}, $U_{población}$ = [0,200], $S_{población}$>

donde la función $S_{población}$ es:

$S_{población}(nadie)=\mu_{(nadie)}$:

$$\mu_{(nadie)}(x) = \begin{cases} 1 & si\, x = 0 \\ 0 & si\, x > 0 \end{cases}$$

$S_{población}(pocos)=\mu_{(pocos)}$:

$$\mu_{(pocos)}(x) = \begin{cases} 1 & si\, x < 2 \\ 1 - ((x-2)/(5-2)) & si\, 2 \le x < 5 \\ 0 & si\, x \ge 5 \end{cases}$$

$S_{población}(algunos)=\mu_{(algunos)}$:

$$\mu_{(algunos)}(x) = \begin{cases} 0 & si\, x < 2 \\ (x-2)/(5-2) & si\, 2 \le x \le 5 \\ 1 - (x-5)/(15-5) & si\, 5 \le x \le 15 \\ 0 & si\, x > 15 \end{cases}$$

$S_{población}(muchos)=\mu_{(muchos)}$:

$$\mu_{(muchos)}(x)=\begin{cases}0 & si \quad x\leq 5\\ (x-5)/(15-5) & si \quad 5<x\leq 15\\ 1-(x-15)/(35-15) & si \quad 15<x\leq 35\\ 0 & si \quad x>35\end{cases}$$

$S_{población}$(demasiados)=$\mu_{(demasiados)}$:

$$\mu_{(demasiados)}(x)=\begin{cases}0 & si \quad x\leq 15\\ (x-15)/(35-15) & si \quad 15<x\leq 35\\ 1 & si \quad x>35\end{cases}$$

La siguiente ilustración muestra representación gráfica de estas funciones de pertenencia:

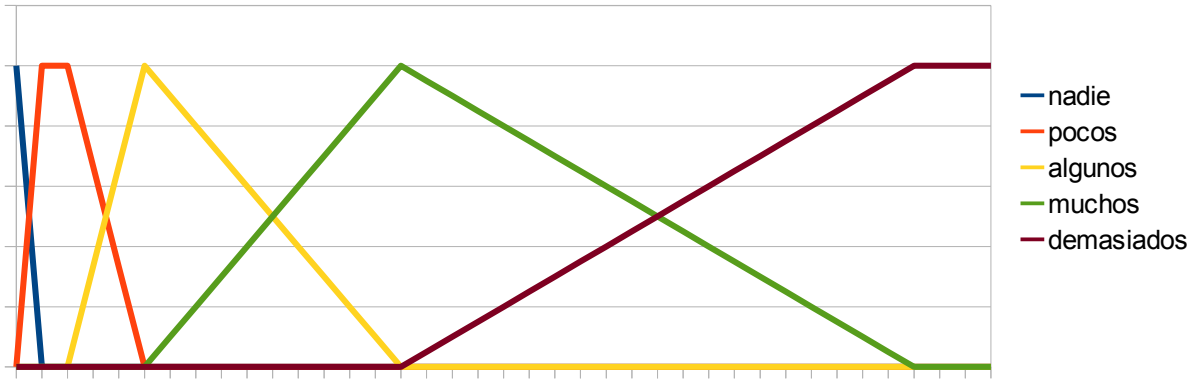


Ilustración 36: Representación gráfica de las funciones de pertenencia μ(nadie), μ(pocos), μ(algunos), μ(muchos), μ(demasiados).

Definición de la variable lingüística de entrada estadoi:

<Estadoi, $L_{estadoi}$ = {muy triste, triste, indiferente, alegre, muy alegre}, $U_{estadoi}$ = [0,2], $S_{estadoi}$>

donde la función $S_{estadoi}$ es:

$S_{estadoi}$(mtriste)=$\mu_{(mtriste)}$:

$$\mu_{(mtriste)}(x)=\begin{cases}1 & si \quad x=0\\ 1-((x-0{,}2)/(0{,}2-0)) & si \quad 0\leq x\leq 0{,}2\\ 0 & si \quad x>0{,}2\end{cases}$$

$S_{estadoi}$(triste)=$\mu_{(triste)}$:

$$\mu_{(triste)}(x)=\begin{cases}0 & si \quad x<0{,}1\\ (x-0{,}2)/(0{,}2-0) & si \quad 0\leq x\leq 0{,}2\\ 1-(x-0{,}1)/(0{,}8-0{,}1) & si \quad 0{,}1\leq x\leq 0{,}8\\ 0 & si \quad x>0{,}8\end{cases}$$

$S_{estadoi}$(indiferente)=$\mu_{(indiferente)}$:

$$\mu_{(indiferente)}(x)=\begin{cases}0 & si \quad x<0,1\\ (x-0,1)/(0,8-0,1) & si \quad 0,1\le x\le 0,8\\ 1-(x-0,7)/(1,3-0,7) & si \quad 0,7\le x\le 1,3\\ 0 & si \quad x>1,3\end{cases}$$

$S_{estadoi}$(alegre)=$\mu_{(alegre)}$:

$$\mu_{(alegre)}(x)=\begin{cases}0 & si \quad x\le 1,2\\ (x-0,7)/(1,3-0,7) & si \quad 0,7<x\le 1,3\\ 1-(x-1,3)/(1,7-1,3) & si \quad 1,3<x\le 1,7\\ 0 & si \quad x>1,7\end{cases}$$

$S_{estadoi}$(malegre)=$\mu_{(malegre)}$:

$$\mu_{(malegre)}(x)=\begin{cases}0 & si \quad x\le 1,8\\ (x-0,7)/(1,7-1,3) & si \quad 1,3<x\le 1,7\\ 1 & si \quad x>1,9\end{cases}$$

La siguiente ilustración muestra representación gráfica de estas funciones de pertenencia:

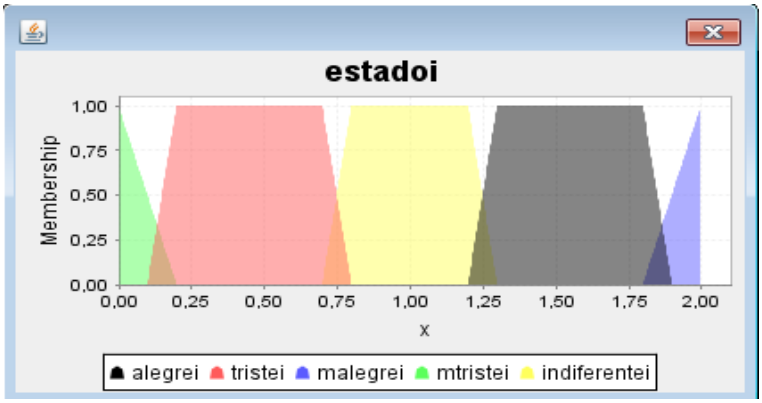
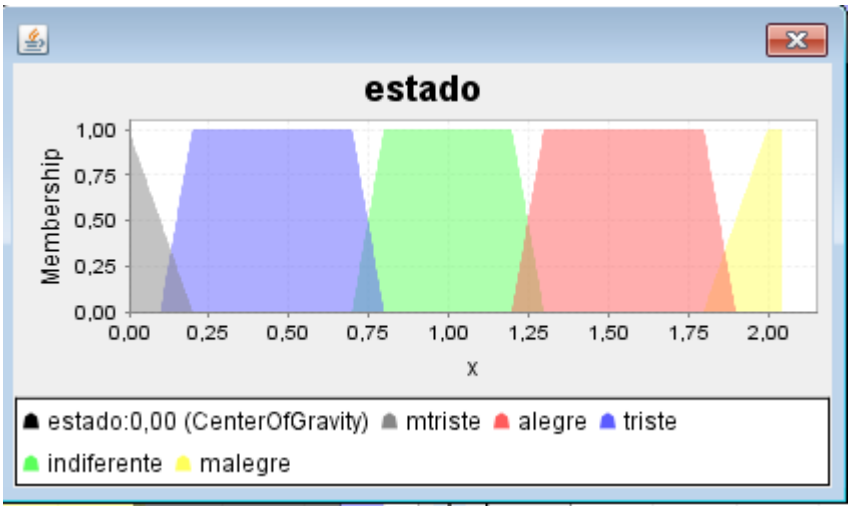


Ilustración 37: Representación gráfica de las funciones de pertenencia de la variable de entrada estado inicial de Tito

Definición de la variable lingüística de salida estado:

<Estado, L_{estado} = {muy triste, triste, indiferente, alegre, muy alegre}, U_{estado} = [0,2], S_{estado}>

donde la función S_{estado} es igual que la definida para función $S_{estadoi}$



Base de Reglas:

Las reglas aplicadas en los sistemas que emplean la lógica difusa son solo aproximadas que pueden variarse con la experiencia o añadiendo sistemas de aprendizaje.

```
RULE 1 : IF poblacion IS nadie AND estadoi IS mtristei THEN estado IS mtriste;
RULE 2 : IF poblacion IS nadie AND estadoi IS tristei THEN estado IS mtriste;
RULE 3 : IF poblacion IS nadie AND estadoi IS indiferentei THEN estado IS triste;
RULE 4 : IF poblacion IS nadie AND estadoi IS alegrei THEN estado IS indiferente;
RULE 5 : IF poblacion IS nadie AND estadoi IS malegrei THEN estado IS alegre;
RULE 6 : IF poblacion IS pocos AND estadoi IS mtristei THEN estado IS mtriste;
RULE 7 : IF poblacion IS pocos AND estadoi IS tristei THEN estado IS triste;
RULE 8 : IF poblacion IS pocos AND estadoi IS indiferentei THEN estado IS indiferente;
RULE 9 : IF poblacion IS pocos AND estadoi IS alegrei THEN estado IS indiferente;
RULE 10: IF poblacion IS pocos AND estadoi IS malegrei THEN estado IS alegre;
RULE 11: IF poblacion IS algunos AND estadoi IS mtristei THEN estado IS mtriste;
RULE 12: IF poblacion IS algunos AND estadoi IS tristei THEN estado IS triste;
RULE 13: IF poblacion IS algunos AND estadoi IS indiferentei THEN estado IS alegre;
RULE 14: IF poblacion IS algunos AND estadoi IS alegrei THEN estado IS alegre;
RULE 15: IF poblacion IS algunos AND estadoi IS malegrei THEN estado IS malegre;
RULE 16: IF poblacion IS muchos AND estadoi IS mtristei THEN estado IS indiferente;
RULE 17: IF poblacion IS muchos AND estadoi IS tristei THEN estado IS alegre;
RULE 18: IF poblacion IS muchos AND estadoi IS indiferentei THEN estado IS malegre;
RULE 19: IF poblacion IS muchos AND estadoi IS alegrei THEN estado IS malegre;
RULE 20: IF poblacion IS muchos AND estadoi IS malegrei THEN estado IS malegre;
RULE 21: IF poblacion IS demasiados AND estadoi IS mtristei THEN estado IS mtriste;
RULE 22: IF poblacion IS demasiados AND estadoi IS tristei THEN estado IS mtriste;
RULE 23: IF poblacion IS demasiados AND estadoi IS indiferentei THEN estado IS triste;
RULE 24: IF poblacion IS demasiados AND estadoi IS alegrei THEN estado IS triste;
RULE 25: IF poblacion IS demasiados AND estadoi IS malegrei THEN estado IS alegre;
```

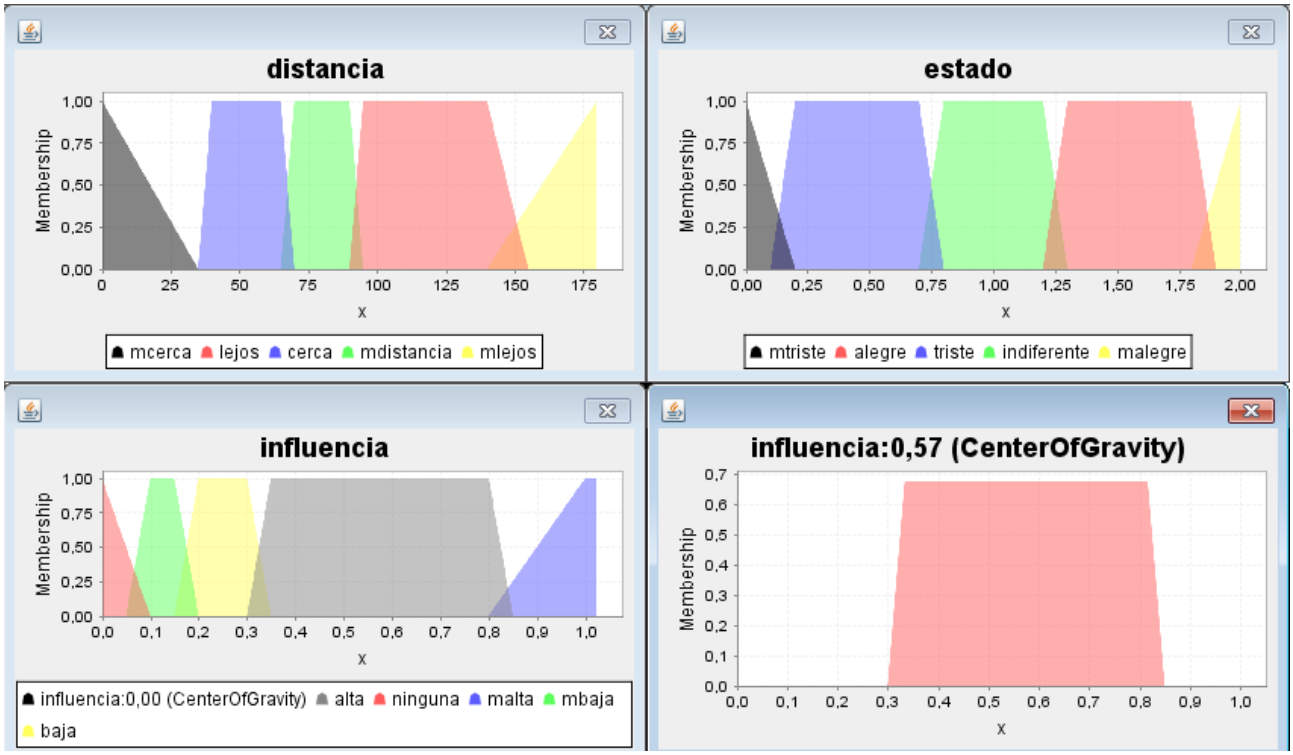
Las reglas utilizadas para el prototipo Tito son las siguientes:

Población / estado inicial	nadie	pocos	algunos	muchos	demasiados
mtristei	mtriste	mtriste	mtriste	indiferente	mtriste
tristei	mtriste	triste	triste	alegre	mtriste
indiferentei	triste	indiferente	alegre	malegre	triste
alegrei	indiferente	indiferente	alegre	malegre	triste
malegrei	alegre	alegre	malegre	malegre	alegre

5.1.3. Evaluación de la percepción:

Para este módulo del sistema también se utiliza la lógica difusa.

Los valores que se obtienen del entorno son para cada personaje, su estado anímico y la distancia a la que se encuentre el personaje del centro de la escena dónde se encuentra el agente inteligente. En función de estos dos valores se obtiene un valor de la influencia sobre el estado de ánimo de Tito.



La suma de este valor de todos los personajes en escena será la variable de entrada población del motor de inferencia del sistema.

En el anexo 2 se presenta el fichero fcl que se utiliza para evaluar la percepción para obtener la población.

5.2. Diseño del entorno virtual y su implementación en VRML .

La codificación del mundo se ha hecho completamente a mano con un editor de texto.

El entorno diseñado está compuesto por la escena estática (cielo y tierra) que adquiere diferentes tonalidades con el paso del tiempo para simular las diferentes estaciones del año. También contiene la representación del agente virtual inteligente a desarrollar para el proyecto que he llamado Tito, éste está representado con una forma humana muy simple mediante formas geométricas básicas del lenguaje VRML (dos cilindros y una esfera) que adquieren inicialmente un color amarillo, representando un estado de ánimo indiferente pero dotándolo de la característica de cambiar su color.

A la escena con el paso del tiempo se la van añadiendo personajes (hombres y mujeres) que se posicionan en la escena de manera aleatoria y que de la misma manera adquieren un color determinado y un movimiento también aleatorio o sin movimiento.

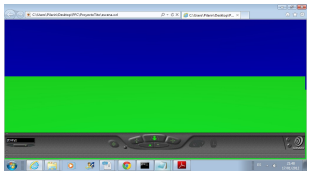
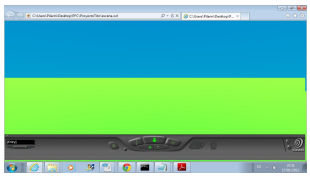
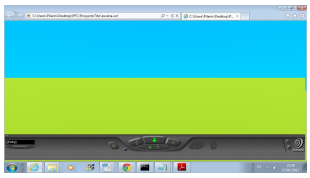
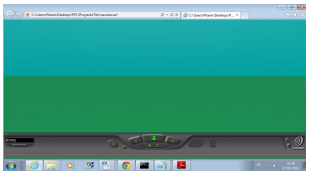
Todos estos comportamientos se implementan en un script interno en el mismo fichero VRML.

5.2.1. El escenario.

Para el escenario se han utilizado 4 *nodos de entorno* de tipo *Background*, uno por cada estación del año. Los *nodos de entorno* tienen la función de modificar el entorno del usuario, para ello utilizan una pila (stack) para cada tipo de nodo, de manera que definiendo diferentes nodos *Background* se pueden ir alternando estos, puesto que solo uno de los nodos de la pila estará activo.

El nodo *Background* permite definir el fondo de la escena bien por medio de una esfera conteniendo una gradación de colores, o bien mediante seis imágenes, una para cada cara interna del cubo que envuelve el mundo virtual.

Se escogió la primera opción, la de utilizar colores, diferenciando el cielo del suelo. Las siguientes figuras muestran las 4 estaciones:

Mediante *ROUTE* y un nodo *Script* se van alternando los cambios de estación.

```
ROUTE EstacionTIMER.cycleTime TO CambioEstacion.cicloEstacion
```

Para activar el script se utiliza un nodo TimeS*ensor.*

```
# Sensor del tiempo para controlar las estaciones del año
DEF EstacionTIMER TimeSensor {
        cycleInterval 90
        loop TRUE
}
```

El script de define de la siguiente manera:

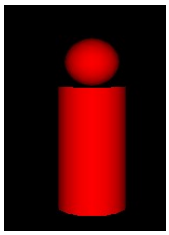
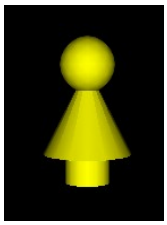
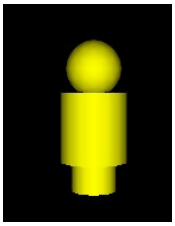
```
DEF CambioEstacion Script {
        # Declarations of what's in this Script node:
        eventIn SFTime cicloEstacion

        field SFBool nueva FALSE
        field SFNode node1 USE prima
        field SFNode node2 USE vera
        field SFNode node3 USE oto
        field SFNode node4 USE inver
        directOutput TRUE
  # Implementation of the logic:
  url "javascript:
     function cicloEstacion(value) {
                nueva = TRUE;
                if (node1.isBound) node2.set_bind = TRUE;
                else if (node2.isBound)      node3.set_bind = TRUE;
                        else if (node3.isBound) node4.set_bind = TRUE;
                                else node1.set_bind = TRUE;
        }"
}
```

El campo *set.bind* de un nodo *Background* coloca ese nodo en la cima de la pila, es decir que lo hace visible.

El campo *isBound* de un nodo *Background* indica si es o no el nodo activo.

5.2.2. Los personajes en escena.

Existen dos tipos de personajes en el mundo virtual de Tito; Tito que representa a un agente autónomo inteligente y otras personas (hombres y mujeres) que carecen de autonomía e inteligencia, a pesar de mostrar cada personaje una animación propia.

Para diseñar los personajes se ha utilizado el nodo *group,* el nodo *group* cumple la función de agrupar diferentes nodo en un mismo nivel de jerarquía, de esta manera las acciones y propiedades que se le apliquen son uniformes en todos los nodos que lo componen. Dentro del campo *children* se especifica el conjunto de nodos del grupo.

```
Group {
        children[
                Shape {
                        geometry Sphere { radius 0.5}
                        appearance Appearance {
                                material Material { diffuseColor 1 1 0}
                        }
                }
                Transform{
                        translation 0 -.8 0
                        children [
                                Shape {
                                        geometry Cone {bottomRadius .8 height 1.5}
                                        appearance Appearance {
                                                material Material { diffuseColor 1 1 0}
                                        }
                                }
                        ]
                }
                Transform{
                        translation 0 -1.7 0
                        children [
                                Shape {
                                        geometry Cylinder {radius 0.4 height .7}
                                        appearance Appearance {
                                                material Material { diffuseColor 1 1 0}
                                        }
                                }
                        ]
                }
        ]
}
```

5.2.3. Comportamientos en el mundo virtual de Tito.

En un apartado anterior se ha explicado como se escenifica los cambios de estaciones mediante la pila del nodo Background. En este apartado se va a describir el comportamiento de los personajes inmersos en la escena.

Inicialmente el entorno únicamente contiene a Tito con un estado emocional muy triste, al encontrarse completamente solo. A medida que pasa el tiempo el entorno se va llenando con mujeres y hombres de manera aleatoria.

En concreto los personajes se crean desde un script mediante un método de la clase Browse.

```
Browser.createVrmlFromString( String )
```

Dicho script (AñadeGente) se activa cada 10 ciclos de tiempo, generando cada vez un nuevo personaje o eliminando un personaje de la escena. Los datos de los personajes, el color y la posición inicial así como los datos de qué personaje se elimina se van apareciendo en la consola de Cosmo Player para posteriormente almacenarlos en un fichero *log* que será recuperado por el programa *AgenteTito.class.*

Algunos personajes se crean con un movimiento aleatorio, para ello se utiliza nodos interpoladores de posición, asignándoles aleatoriamente unas posiciones de inicio, medio camino y fin, y un ciclo determinado de manera aleatoria para inducirles una velocidad determinada.

```
PROTO chica [
        field SFColor color 1 1 0
        field SFVec3f posicion 0 0 0
        field MFVec3f valorK [30 0 50, -30 0 30, 30 0 50]
]
{
        DEF tr Transform {
                translation IS posicion
                children [
                        DEF pi PositionInterpolator {
                                key [ 0 .5 1 ]
                                keyValue IS valorK
                        }

                        DEF ts TimeSensor {
                                cycleInterval IS ciclo
                                loop TRUE
                        }

                        Group {
                                children[
                                Shape {
                                        geometry Sphere { radius 0.5}
                                                appearance Appearance {
                                                material Material { diffuseColor IS color }
                                        }
                                }
                                Transform{
                                        translation 0 -.8 0
                                        children [
                                                Shape {
                                                        geometry Cone {bottomRadius .8 height 1.5}
                                                        appearance Appearance {
                                                                material Material { diffuseColor IS color}
                                                        }
                                                }
                                        ]
                                }
                                Transform{
                                        translation 0 -1.7 0
                                        children [
                                                Shape {
                                                        geometry Cylinder {radius 0.4 height .7}
                                                        appearance Appearance {
                                                                material Material { diffuseColor IS color}
                                                        }
                                                }
                                        ]
                                }
                                ]
                        }

                ]
        }
        ROUTE ts.fraction_changed TO pi.set_fraction
        ROUTE pi.value_changed TO tr.set_translation

}
```

Las variables color, posicion, valorK y ciclo, se calculan dentro del script de manera aleatoria.

Cada cierto tiempo (120 ciclos) se carga de nuevo el personaje Tito variando así el color que representa su nivel de alegría. El fichero desde donde se carga es Tito.wrl que es actualizado externamente desde AgenteTito.class.

```
DEF CambiaColorTito Script {
        eventIn SFTime ciclo
        field SFNode gente USE gente

        directOutput TRUE

        # Implementation of the logic:
        url "javascript:
                function ciclo(Value) {
                        urlString = new MFString('Tito.wrl');
                        Nuevo = Browser.createVrmlFromURL(urlString, gente,'addChildren');
                }"
}
```

5.2.4. Interacción con el entorno.

La interacción con el entorno es mínima, el usuario solo podrá cambiar el punto de vista de entre los cuatro que se han implementado:

- Punto de vista principal, a una distancia del centro de la escena donde se encuentra Tito de 100
- Vista de pájaro, para ver la escena desde el cielo.
- Cerca de Tito, para acercarnos a una distancia de 20.
- Vista de Tito, que sitúa el punto de vista del usuario sobre un hombro de Tito.

Los puntos de vista se pueden ir alternando con las teclas AvPág y RePág.

5.3. Implementación del prototipo en Java.

5.3.1. Visión general.

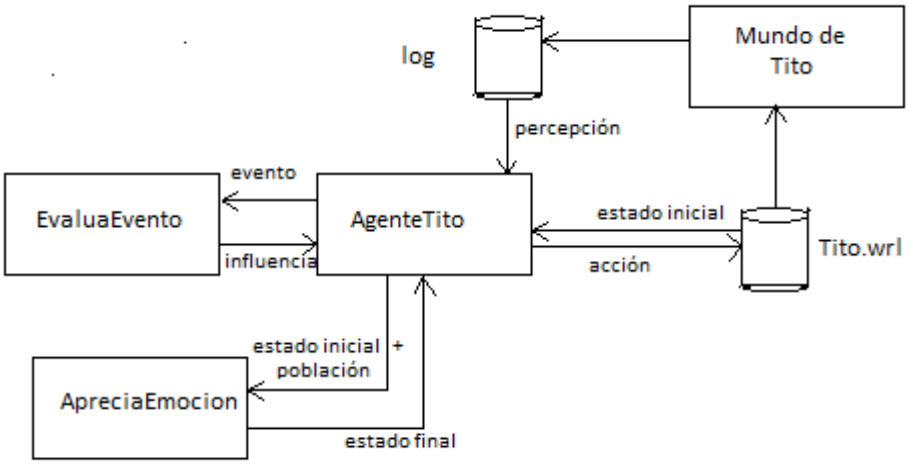


Ilustración 38: Visión general del prototipo implementado

5.3.2. AgenteTito.java

Este el el componente principal, se encarga de leer los datos del fichero log que se han obtenido del mundo de Tito, estos datos representan la percepción del agente inteligente de su entorno durante un periodo de tiempo determinado.

Los datos son almacenados una tabla que se envían al módulo EvaluaEvento que devuelve como resultado la variable de entrada (población) del motor de inferencia del sistema inteligente.

También lee el estado anterior al periodo de tiempo evaluado desde el fichero Tito.wrl.

Estos datos son tratados por el módulo apreciaEmoción, dando como resultado el nuevo estado de ánimo de Tito que es actualizado en el fichero Tito.wrl.

Este fichero Tito.wrl es el fichero que utiliza el mundo de Tito para actualizar el estado de ánimo de Tito representado por el color cada 120 ciclos de tiempo.

5.3.3. EvaluaEvento.java

EvaluaEvento calcula la distancia y el estado de ánimo de todos los personajes en escena. Para calcular la distancia se utiliza la siguiente fórmula:

$$D(P,Q) = \sqrt{(p_x - q_x)^2 + (p_z - q_z)^2}$$

El estado de ánimo es un valor entre 0 y 2 que se calcula mediante la siguiente fórmula.

$$estado = \begin{cases} 1 - valorRed & si \quad valorRed > valorGreen \\ 1 & si \quad valorRed = valorGreen \\ valorGreen + 1 & si \quad valorRed < valorGreen \end{cases}$$

Mediante las reglas contenidas en el fichero evalua.fcl calcula la influencia de cada personaje.

La influencia es un valor entre 0 y 1 que depende de lo lejos que está el personaje de Tito y de su estado de ánimo. Sumando todas las influencias de todos los personajes se obtiene un valor de población que servirá

para inferir un nuevo estado de ánimo a Tito.

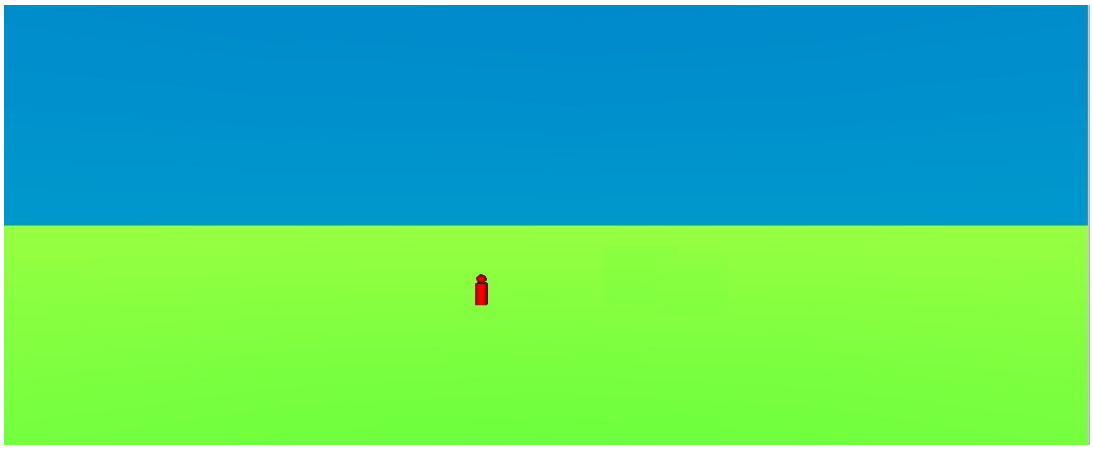
5.3.4. ApreciaEmocion.java

En este módulo devolverá el resultado de la inferencia, utilizando el fichero aprecia.fcl que contiene las reglas de inferencia.

5.4. Pruebas realizadas con el prototipo.

5.4.1. Estado inicial.

Inicialmente Tito está muy triste porque no hay ningún personaje en escena. Su estado se representa con el color rojo.



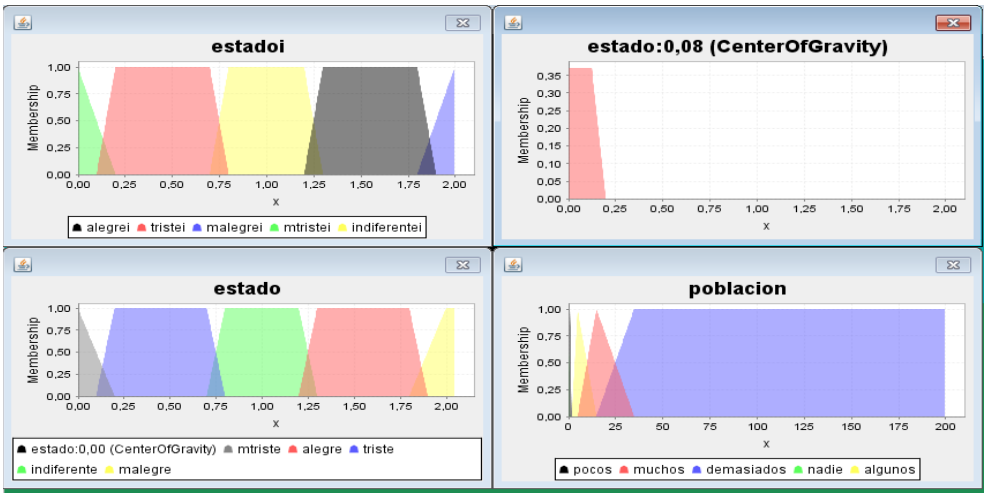
5.4.2. Un ejemplo de evaluación.

Durante 60 intervalos de tiempo, aparecen en escena 6 personajes

intervalos	distancia	estado	influencia
10	104.69479452198185	0.55833685769216	0.03300000000000001
20	103.87011119662864	0.36524367232905697	0.03300000000000001
30	55.08175741568164	0.7695063961793339	0.5362243687939192
40	74.33034373659252	1.386360456790867	0.933
50	37.73592452822641	0.6828963740027311	0.2500000000000004
60	59.36328831862332	1.256333158299566	0.6627944607554425

Lo representa una población de 3,110813290304805.

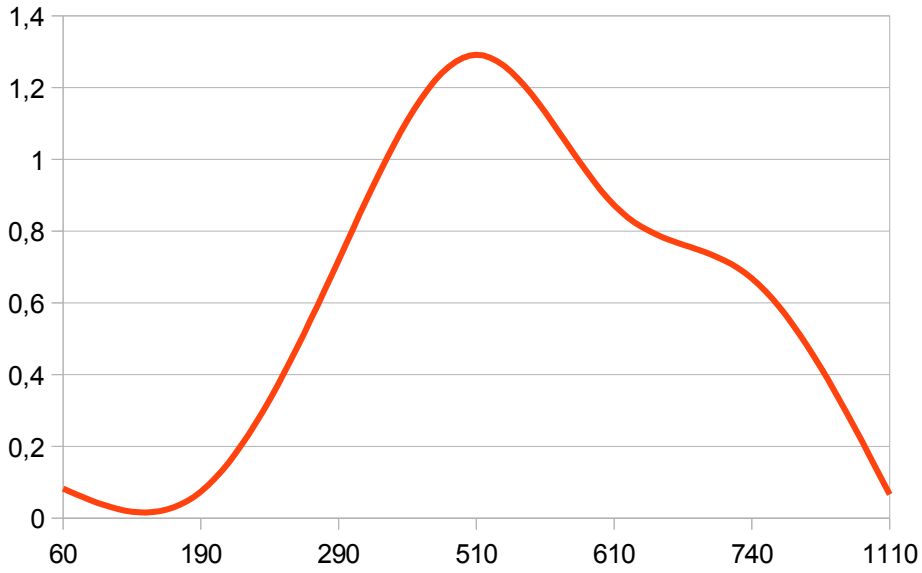
Una vez evaluado el sistema Tito cambia levemente de estado a {0.9176212277856025 0 0} cuando inicialmente era {1 0 0}.



5.4.3. Resultados

Se toman datos en los intervalos 60, 190, 290, 510, 610, 740 y 1110, con los siguientes resultados.

	población	estadoi	estado final
60	3.110813290304805	0	0.08237877221439747
190	4.538741669519483	0.08237877221439747	0.07414744855151094
290	8.543613915772736	0.07414744855151094	0.7192488183318225
510	19.230909252238195	0.7192488183318225	1.2914078035302101
610	21.618499369221546	1.2914078035302101	0.8724439840997023
740	26.687467856910686	0.8724439840997023	0.668207275398435
1110	40.32661746381641	0.668207275398435	0.06600000000000002



Los resultados demuestran que en el motor de inferencia que se ha utilizado el factor que más se tiene en cuenta para modificar el estado de ánimo del agente es la población. Entre los valores 8 y 19 de población el estado de ánimo es creciente, a partir del valor 19 el estado de ánimo empeora.

6. CONCLUSIONES Y LINEAS FUTURAS.

Para implementar este prototipo de un entorno virtual inteligente, se tomó la decisión al principio del proyecto de utilizar como herramienta de desarrollo el Lenguaje de modelado de mundos virtuales VRML, por la facilidad de implementación y por soportar la comunicación externa con un lenguaje como Java en cualquier plataforma. Pero a medida que se fue avanzando en el proyecto se ha visto que no ha sido la mejor opción. Esa comunicación con Java mediante el EAI del Cosmo Player ha resultado estar obsoleta para las versiones actuales de Java y también con los navegadores.

Con X3D también es posible esa comunicación con lenguajes externos para el desarrollo de aplicaciones más complejas, ha sustituido prácticamente a VRML, por este motivo no se han desarrollado actualizaciones de EAI de Cosmo Player.

Pero el desarrollo de X3D está más limitado en Windows. Por eso a pesar de que un entorno VRML es fácilmente portable a X3D, el cambio a mitad de proyecto hubiera supuesto un cambio de toda la plataforma elegida al principio para su desarrollo, desde el sistema operativo hasta el visualizador.

Se intentó solucionar el problema cambiando el visualizador, en concreto utilizando FreeWRL, puesto que se está desarrollando y en teoría es compatible con los dos estándares VRML y X3D. El problema es que está desarrollado inicialmente para Linux, y aunque hay una versión para Windows ésta no funciona.

Salvando estos problemas se ha podido implementar un prototipo, obviando la comunicación directa del agente con el entorno virtual, que incluye un comportamiento emocional no verbal a un agente inmerso en un mundo virtual.

El prototipo se ha implementado basándose en parte del modelo FLAME, por lo que un aspecto a mejorar en el motor de inferencia, sería, por ejemplo, tener en cuenta el decaimiento de la emoción dependiendo del tiempo que se mantiene el mismo estado de ánimo.

Con respecto al diseño del entorno y la forma de representación de la emoción de alegría y los personajes, hay muchos aspectos a mejorar, pero el principal es la utilización de otras herramientas que permitan una relación entorno-agente directa y ágil. En mi opinión, la opción escogida pasaría sin duda por utilizar como sistema operativo Linux y las herramientas de software libre que se están desarrollando.

El desarrollo de este tipo de Mundos Virtuales debe basarse en la interoperabilidad y la estandarización en plataformas de código abierto, por lo que cualquier estudio relacionado con esta área debería seguir estos criterios.

Por otro lado, a pesar de que con este proyecto no se pretendía ninguna aplicación concreta, dotar de inteligencia a los entornos virtuales es fundamental, y en concreto el reconocimiento de emociones tiene especial importancia sobretodo en el área de sistemas de interacción humana ya que permite mejorar la calidad de los servicios prestados por estos sistemas, habilitándolos para tomar decisiones basándose en el estado emocional de los usuarios. Reconocer el estado de ánimo de los usuarios brinda información

relevante al sistema, retroalimentándolo y haciéndolo capaz de reaccionar y adaptarse.

Las aplicaciones de este tipo de sistemas inteligentes son infinitas: desde tutoriales interactivos en el que se podría motivar y captar el interés dependiendo del estado emocional del alumno, hasta sistemas telefónicos de atención automática para asistencia médica que diera prioridad a las llamadas más urgentes que se podrían diferenciar dependiendo del estado emocional del usuario.

7. GLOSARIO DE TÉRMINOS.

MMO: Massively multiplayer online games.

MMOLE: Massively Multiple On Line Education.

MUD: Multi User Dungeon (Mazmorra/Calabozo Multiusuario) Es un juego de rol online multijugador de solo texto. Este tipo de juegos se crearon principalmente en la década de los 70 cuando eran prácticamente inexistentes los videojuegos gráficos, y con el tiempo estos han evolucionando hasta los actuales MMORPG. Para jugar a estos juegos ni siquiera era necesario un programa especifico, puesto que se podía jugar a ellos simplemente utilizando cualquier programa que soportaría el protocolo Telnet.

MMORPG: Massive Multiplayer Online Role Playing Game (Juego de Rol Online Multijugador Masivo) Algunos de los más populares en la actualidad son World of Warcraft, Lineage 2, Tibia...

Telepresencia: Esta es una variación de mundos generados por computador, pues se conectan sensores remotos ubicados en el mundo real, estos sensores se hallan generalmente en un robot equipado con cámaras que facilitan al usuario su orientación y destreza. Este sistema de realidad virtual mezcla el mundo real con el virtual, con el fin de que el usuario interactúe en ambos mundos.

VRML: Virtual Reality Modelling Language (Lenguaje Modelado Realidad Virtual) Formato de fichero estandarizado para representar imágenes o mundos 3D. Este formato fue diseñado para ser utilizado en la web. La primera versión del mismo fue especificada en 1994, y en 1997 una nueva versión llamada VRML2 o VRML97 fue presentada. Actualmente está en desuso y ha sido sustituido por el X3D.

Metaverso: Los metaversos son entornos donde los humanos interactúan social y económicamente como iconos a través de un soporte lógico en un ciberespacio que representa una metáfora del mundo real, pero sin las limitaciones físicas.

8. FUENTES DE INFORMACIÓN Y BIBLIOGRAFÍA

[1] Ruth Aylett, Michael Luck; ,"Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments",-2000-

[2] 25-oct-2011,"Noticias actuales sobre tecnología e informática",Julio-2007- http://www.tendencias21.net/TENDENCIAS-INFORMATICAS_r22.html

[3] Pilar Herrero, Angélica de Antonio, ,"Diseño de un modelo de percepción para Agentes Virtuales Inteligentes basado en el sistema de percepción de los seres humanos",,-2003-

[4] Wooldridge, M. J. and Jennings, N.R.,"Intelligent agents: Theory and practice", ,--

[5] Demetri Terzopoulos, Tamer Rabie and Radek Grzeszczuk, ,"Perception and Learning in Artificial Animals", ,-1996-

[6] J-S. Monzani, A. Caicedo, and D. Thalmann, ,“Integrating Behavioural Animation Techniques”, ,-2001-

[7] P. Becheiraz and D. Thalmann, ,“A Behavioral Animation System for Autonomous Actors Personified by Emotions”, ,-1998-

[8] M. Minsky, The Society of the Mind, 1986,New York: Simon & Schuster,

[9] Stacy Marsella, Jonathan Gratch, Paolo Petta,"Computational Models of Emotion",-2010-

[10] 25-oct-2011,"Desarrollando personajes virtuales con psicología humana", ,marzo-2008- http://www.solociencia.com/informatica/08042303.htm

[11] 10-oct-2011,The Virtual Reality Modeling Language,-- http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/

[12] 20-nov-2011,"Extensible 3D (X3D): Architecture and base components",-- http://www.web3d.org/x3d/specifications/ISO-IEC-19775-1.2-X3D-AbstractSpecification/

[13] 5-dic-2011, "Fuzzy Control Programming", ,,

[14] ,"JFuzzyLogic Package",-- http://jfuzzylogic.sourceforge.net/doc/net/sourceforge/jFuzzyLogic/package-summary.html

[15] Magy Seif El-Nasr , John Yen, Thomas R. Ioerger,"FLAME - Fuzzy Logic Adaptive Model of Emotions",-2000-

9. ANEXOS.

9.1. Anexo A: Sistema de inferencia del estado de ánimo de Tito.

```
/*
        Example: Calculo del estado de Tito con FIS (fuzzy inference system)
        Calcula el estado basándose en la cantidad de personas que le rodean y el estado de ánimo en que se encuentra Tipo en un momento dado

*/


FUNCTION_BLOCK estadot

VAR_INPUT                               // Define input variables
        poblacion : REAL;
        estadoi: REAL;
END_VAR

VAR_OUTPUT                              // Define output variable
        estado: REAL;
END_VAR

FUZZIFY poblacion                       // Fuzzify input variable 'poblacion' : {'nadie', 'pocos', 'algunos', 'muchos', 'demasiados' }
        TERM nadie := (0, 1) (1, 0);
        TERM pocos := (0,0) (1, 1) (2, 0);
        TERM algunos := (2, 0) (5, 1) (15, 0);
        TERM muchos := (5, 0) (15, 1) (35, 0);
        TERM demasiados := (15, 0) (35, 1) (200, 1);
END_FUZZIFY

FUZZIFY estadoi                         // Fuzzify input variable 'estadoi': {'mtriste', 'triste' , 'indiferente', 'alegre', 'malegre' }
        TERM mtristei := (0, 1) (0.2, 0) ;
        TERM tristei := (0.1, 0) (0.2, 1) (0.7, 1) (0.8, 0);
        TERM indiferentei := (0.7, 0) (0.8, 1) (1.2, 1) (1.3, 0);
        TERM alegrei := (1.2, 0) (1.3, 1) (1.8, 1) (1.9, 0);
        TERM malegrei := (1.8, 0) (2, 1);
END_FUZZIFY

DEFUZZIFY estado                        // Defuzzify output variable 'estado': {'mtriste', 'triste' , 'indiferente', 'alegre', 'malegre' }
        TERM mtriste := (0, 1) (0.2, 0) ;
        TERM triste := (0.1, 0) (0.2, 1) (0.7, 1) (0.8, 0);
        TERM indiferente := (0.7, 0) (0.8, 1) (1.2, 1) (1.3, 0);
        TERM alegre := (1.2, 0) (1.3, 1) (1.8, 1) (1.9, 0);
        TERM malegre := (1.8, 0) (2, 1);
        METHOD : COG;                   // Use 'Center Of Gravity' defuzzification method
        DEFAULT := 0;           // Default value is 0 (if no rule activates defuzzifier)
END_DEFUZZIFY
```

```
RULEBLOCK No1
        AND : MIN;                  // Use 'min' for 'and' (also implicit use 'max' for 'or' to fulfill
DeMorgan's Law)
        ACT : MIN;                  // Use 'min' activation method
        ACCU : MAX;                 // Use 'max' accumulation method
        RULE 1 : IF poblacion IS nadie AND estadoi IS mtristei THEN estado IS mtriste;
        RULE 2 : IF poblacion IS nadie AND estadoi IS tristei THEN estado IS mtriste;
        RULE 3 : IF poblacion IS nadie AND estadoi IS indiferentei THEN estado IS triste;
        RULE 4 : IF poblacion IS nadie AND estadoi IS alegrei THEN estado IS indiferente;
        RULE 5 : IF poblacion IS nadie AND estadoi IS malegrei THEN estado IS alegre;
        RULE 6 : IF poblacion IS pocos AND estadoi IS mtristei THEN estado IS mtriste;
        RULE 7 : IF poblacion IS pocos AND estadoi IS tristei THEN estado IS triste;
        RULE 8 : IF poblacion IS pocos AND estadoi IS indiferentei THEN estado IS indiferente;
        RULE 9 : IF poblacion IS pocos AND estadoi IS alegrei THEN estado IS indiferente;
        RULE 10: IF poblacion IS pocos AND estadoi IS malegrei THEN estado IS alegre;
        RULE 11: IF poblacion IS algunos AND estadoi IS mtristei THEN estado IS mtriste;
        RULE 12: IF poblacion IS algunos AND estadoi IS tristei THEN estado IS triste;
        RULE 13: IF poblacion IS algunos AND estadoi IS indiferentei THEN estado IS alegre;
        RULE 14: IF poblacion IS algunos AND estadoi IS alegrei THEN estado IS alegre;
        RULE 15: IF poblacion IS algunos AND estadoi IS malegrei THEN estado IS malegre;
        RULE 16: IF poblacion IS muchos AND estadoi IS mtristei THEN estado IS indiferente;
        RULE 17: IF poblacion IS muchos AND estadoi IS tristei THEN estado IS alegre;
        RULE 18: IF poblacion IS muchos AND estadoi IS indiferentei THEN estado IS malegre;
        RULE 19: IF poblacion IS muchos AND estadoi IS alegrei THEN estado IS malegre;
        RULE 20: IF poblacion IS muchos AND estadoi IS malegrei THEN estado IS malegre;
        RULE 21: IF poblacion IS demasiados AND estadoi IS mtristei THEN estado IS mtriste;
        RULE 22: IF poblacion IS demasiados AND estadoi IS tristei THEN estado IS mtriste;
        RULE 23: IF poblacion IS demasiados AND estadoi IS indiferentei THEN estado IS triste;
        RULE 24: IF poblacion IS demasiados AND estadoi IS alegrei THEN estado IS triste;
        RULE 25: IF poblacion IS demasiados AND estadoi IS malegrei THEN estado IS alegre;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

9.2. Anexo 2: Reglas de inferencia para el módulo de evaluación.

```
/*
        Example: Calculo de la variable población para utilizar en FIS (fuzzy inference system) Tito2

*/

FUNCTION_BLOCK pobXdisstate

VAR_INPUT                       // Define input variables
        estado: REAL;
        distancia: REAL;
END_VAR

VAR_OUTPUT                      // Define output variable
        influencia: REAL;
END_VAR

FUZZIFY estado                  // Fuzzify input variable 'estado': {'mtriste', 'triste' , 'indiferente',
'alegre', 'malegre' }
        TERM mtriste := (0, 1) (0.2, 0) ;
        TERM triste := (0.1, 0) (0.2, 1) (0.7, 1) (0.8, 0);
        TERM indiferente := (0.7, 0) (0.8, 1) (1.2, 1) (1.3, 0);
        TERM alegre := (1.2, 0) (1.3, 1) (1.8, 1) (1.9, 0);
        TERM malegre := (1.8, 0) (2, 1);
END_FUZZIFY

FUZZIFY distancia               // Fuzzify input variable 'distancia': {'mlejos', 'lejos' , 'mdistancia', 'cerca',
'mcerca' }
        TERM mcerca := (0, 1) (35, 0) ;
        TERM cerca := (35, 0) (40, 1) (65, 1) (70, 0);
        TERM mdistancia := (65, 0) (70, 1) (90, 1) (95, 0);
        TERM lejos := (90, 0) (95, 1) (140, 1) (155, 0);
        TERM mlejos := (140, 0) (180, 1);
END_FUZZIFY

DEFUZZIFY influencia            // Defuzzify output variable 'influencia': {'malta', 'alta' , 'baja', 'mbaja',
'ninguna' }
        TERM ninguna := (0, 1) (0.1, 0) ;
        TERM mbaja := (0.05, 0) (0.1, 1) (0.15, 1) (0.2, 0);
        TERM baja := (0.15, 0) (0.2, 1) (0.3, 1) (0.35, 0);
        TERM alta := (0.3, 0) (0.35, 1) (0.8, 1) (0.85, 0);
        TERM malta := (0.8, 0) (1, 1);
        METHOD : COG;           // Use 'Center Of Gravity' defuzzification method
        DEFAULT := 0;           // Default value is 0 (if no rule activates defuzzifier)
END_DEFUZZIFY

RULEBLOCK No1
        AND : MIN;              // Use 'min' for 'and' (also implicit use 'max' for 'or' to fulfill
DeMorgan's Law)
        ACT : MIN;              // Use 'min' activation method
        ACCU : MAX;             // Use 'max' accumulation method
        RULE 1 : IF distancia IS mlejos AND estado IS mtriste THEN influencia IS ninguna;
        RULE 2 : IF distancia IS mlejos AND estado IS triste THEN influencia IS ninguna;
        RULE 3 : IF distancia IS mlejos AND estado IS indiferente THEN influencia IS ninguna;
        RULE 4 : IF distancia IS mlejos AND estado IS alegre THEN influencia IS mbaja;
```

```
        RULE 5 : IF distancia IS mlejos AND estado IS malegre THEN influencia IS baja;
        RULE 6 : IF distancia IS lejos AND estado IS mtriste THEN influencia IS ninguna;
        RULE 7 : IF distancia IS lejos AND estado IS triste THEN influencia IS ninguna;
        RULE 8 : IF distancia IS lejos AND estado IS indiferente THEN influencia IS mbaja;
        RULE 9 : IF distancia IS lejos AND estado IS alegre THEN influencia IS baja;
        RULE 10: IF distancia IS lejos AND estado IS malegre THEN influencia IS alta;
        RULE 11: IF distancia IS mdistancia AND estado IS mtriste THEN influencia IS ninguna;
        RULE 12: IF distancia IS mdistancia AND estado IS triste THEN influencia IS mbaja;
        RULE 13: IF distancia IS mdistancia AND estado IS indiferente THEN influencia IS baja;
        RULE 14: IF distancia IS mdistancia AND estado IS alegre THEN influencia IS malta;
        RULE 15: IF distancia IS mdistancia AND estado IS malegre THEN influencia IS malta;
        RULE 16: IF distancia IS cerca AND estado IS mtriste THEN influencia IS baja;
        RULE 17: IF distancia IS cerca AND estado IS triste THEN influencia IS baja;
        RULE 18: IF distancia IS cerca AND estado IS indiferente THEN influencia IS alta;
        RULE 19: IF distancia IS cerca AND estado IS alegre THEN influencia IS malta;
        RULE 20: IF distancia IS cerca AND estado IS malegre THEN influencia IS malta;
        RULE 21: IF distancia IS mcerca AND estado IS mtriste THEN influencia IS baja;
        RULE 22: IF distancia IS mcerca AND estado IS triste THEN influencia IS alta;
        RULE 23: IF distancia IS mcerca AND estado IS indiferente THEN influencia IS alta;
        RULE 24: IF distancia IS mcerca AND estado IS alegre THEN influencia IS malta;
        RULE 25: IF distancia IS mcerca AND estado IS malegre THEN influencia IS malta;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

9.3. Manual de instalación.

Para poder probar el proyecto es necesario tener instalado java y el visualizador Cosmo Player (que se adjunta al los ficheros del proyecto.

Una vez instalado el Cosmo Player y descomprimida la carpeta ProyectoTito, hay que modificar el CLASSPATH añadiendo el directorio actual (;.) si no está así configurado.

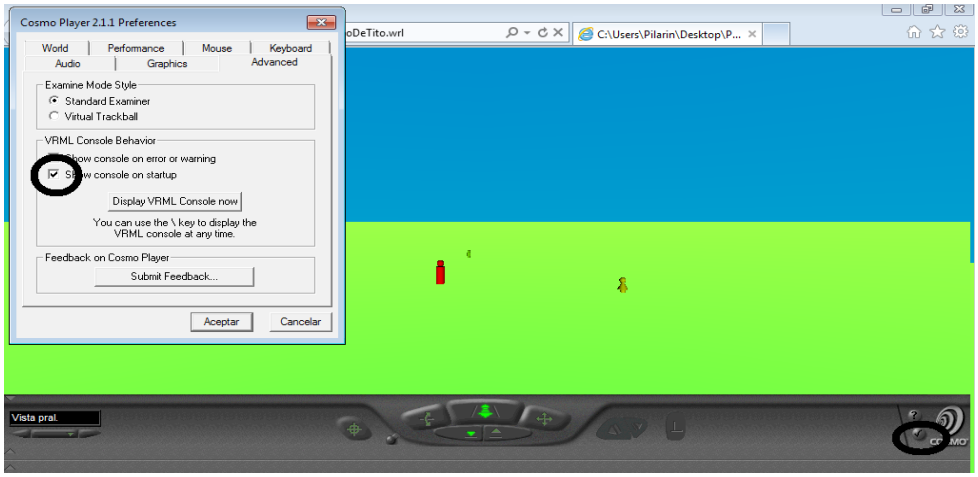
Las carpetas incluidas en ProyectoTito son:

- Fuentes: contiene los ficheros java del proyecto.
- Org: del paquete JfuzzyLogic.
- Net: del paquete JfuzzyLogic.

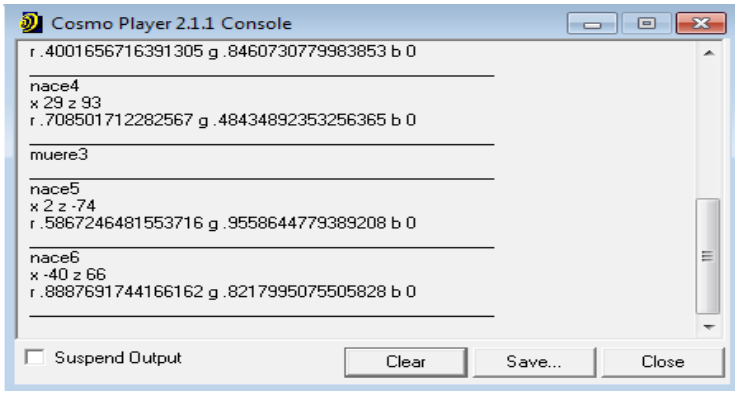
9.4. Manual de usuario.

Para realizar pruebas con el sistema inteligente:

- En primer lugar se deberá abrir el fichero MundoDeTito.wrl con el visualizador CosmoPlayer.
- Una vez abierto se cambiarán las preferencias de Cosmo Player para que se abra la cónsola de CosmoPlayer al abrir cualquier mundo wrl.



- Se vuelve a abrir el fichero MundoDeTito.wrl, y en la cónsola aparecerán los datos que se evaluarán con el agenteTito.class
- Para guardar los datos de la cónsola en el momento que se quiera evaluar se utilizará el comando save de la cónsola y el fichero se nombrará *log*



- Abriendo una cónsola de windows y situados en el directorio ProyectoTito se ejecuta el programa agenteTito.class.
- Los resultados aparecen en la cónsola de windows y además se actualizará en el MundoDeTito.