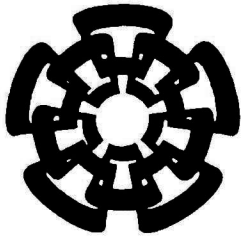
XX(132616.1)



CINVESTAV

Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

Núcleo de GeDA-3D

Tesis que presenta:
Alonso Aguirre Gutierrez

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jalisco, Febrero de 2007.

Núcleo de GeDA-3D

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Alonso Aguirre Gutierrez
Ingeniero Electrónica en Computación
Centro de Enseñanza Técnica Industria 1997-2002

Becario de CONACYT, expediente no. 191399

Director de Tesis
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, Febrero de 2007.

Agradecimientos:

A DIOS

A mis padres,

Mis agradecimientos a mis compañeros y amigos.

A mi asesor,

Al ***CONACYT*** por su apoyo

Becario de CONACYT No. 191399

Resumen:

Esta tesis presenta el trabajo de investigación y desarrollo realizado para implantar el núcleo de una plataforma basada en el paradigma de multi-agente. El núcleo es parte de un proyecto útil para desarrollar aplicaciones distribuidas 3D que consideran ambientes virtuales dinámicos. Este documento expone cada uno de los componentes que conforman el núcleo así como su integración final en el proyecto GeDA-3D. El núcleo presentado permite la evolución de las entidades virtuales y el ambiente donde estás se evolucionan. La plataforma propuesta en este trabajo permite: La interacción entre las entidades lo cual les permite alcanzar los objetivos asignados por el usuario; Mostrar gráficamente los resultados de una escena y que las entidades interactúen dentro del ambiente siguiendo reglas establecidas específicamente para dicho ambiente.

**Palabras clave*: multi-agente, inteligencia artificial distribuida, GeDA-3D.*

Abstract

This thesis presents the work of investigation and development realized to implant the kernel of a platform based on the multi-agent paradigm. The kernel is a part of a project useful to develop distributed 3D applications that take into consideration dynamic virtual environments. This document exposes each one of the components that conform the kernel as well as its final integration into the GeDA-3D project. The presented kernel allows the evolution of the virtual entities and also the environment in which they evolve. The platform proposed in this work allows: The interaction between the entities which allows them to reach the objectives intended by the user; the graphical demonstration of the results of each scene and of the interaction between the entities within the environment following the rules established specifically for this environment.

**Key words*: multi-agent, distributed artificial intelligence, GeDA-3D.*

Índice General

Capítulo 1 Introducción

Resumen

En este capítulo se ofrece una introducción y una reseña sucinta del problema y la solución propuesta, también se presenta una descripción concisa de la estructura de los capítulos de la presente tesis.

Introducción

En el año 2000 nace el proyecto llamado "Arquitectura Genérica de Ambiente Virtual Distribuido" reconocido como GeDA-3D [1.1, 1.2, 1.3]. El proyecto nace con el objetivo de facilitar el desarrollo de aplicaciones distribuidas, el proyecto ha evolucionado y actualmente GeDA-3D es una plataforma basada en el paradigma de agentes de la inteligencia artificial distribuida (IAD), útil para desarrollar aplicaciones distribuidas 3D de una manera amigable (declarativa), por lo tanto puede ser visto como una plataforma útil para diseñar y ejecutar ambientes dinámicos virtuales. En la Figura 1.1 se presenta una arquitectura general del proyecto, en la cual podemos apreciar cada uno de los módulos integrantes del proyecto.

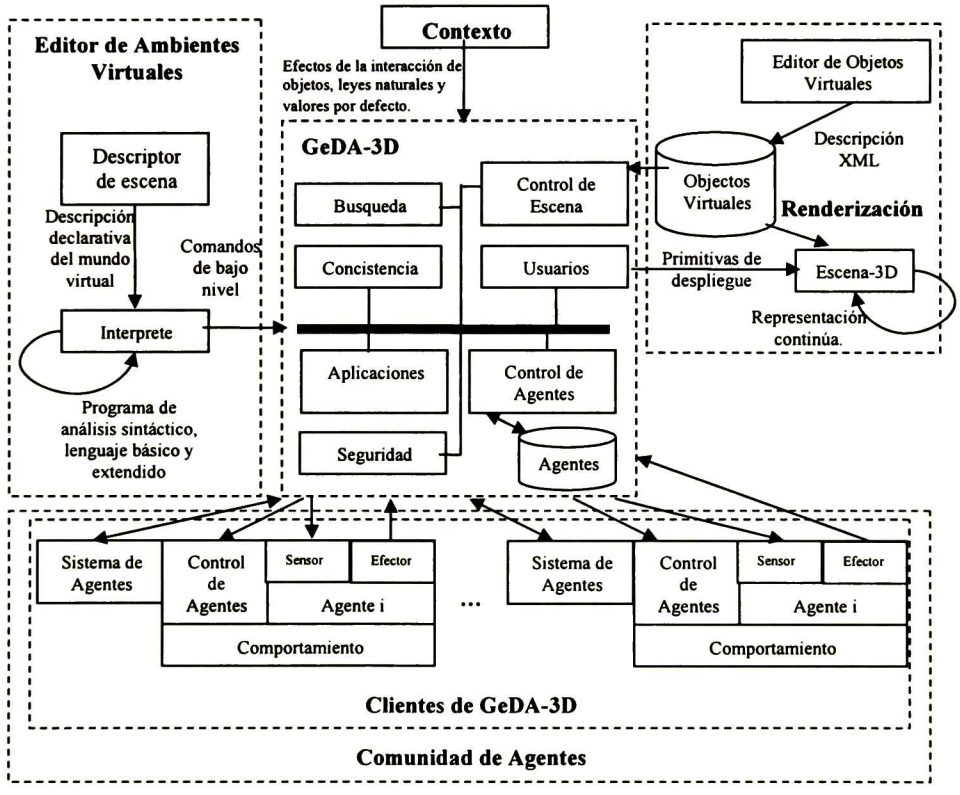


Figura 1.1 Arquitectura General del Proyecto GeDA-3D

Dentro de la arquitectura se puede apreciar los siguientes módulos principales: editor declarativo de ambientes virtuales, editor de contexto de ambientes virtuales,

representador gráfico, comunidad de agentes y núcleo GeDA-3D, descrito a continuación:

- Editor declarativo de ambientes virtuales: Apoya a el usuario del sistema GeDA -3D, fungiendo entre este y la plataforma, este modulo permite al usuario utilizar un lenguaje declarativo simple, para generar el ambiente, describir las características de la personalidad del los entes en la animación, determinar las metas a cumplir por cada una de las entidades en el ambiente, así como las restricciones y ubicaciones de las mismas al comienzo de la simulación. También es el encargado de determinar si la escena que se desea generar es valida.

- Editor de contexto de ambientes virtuales: Su principal función es determinar las siguientes propiedades del ambiente:
    - Los efectos de las colisiones entre todos los diferentes tipos de entes en el ambiente.
    - Las consecuencias de la ejecución de una acción sobre cualquier entidad del ambiente.
    - Las leyes que imperaran en el mundo modelado tales como ley de gravedad, entre otras, reglas que determinan la manera en la que se rige el accionar en el mundo virtual modelado.
    - Las reglas que limitan la generación de entidades de acuerdo a la escena descrita.

- Representador gráfico: Es la carta de presentación del proyecto, se encarga de representar la escena y ambiente descritos, así como a cada una de las entidades dentro del ambiente virtual en un ámbito grafico, también representa las acciones y reacciones de cada uno de los avatares y objetos visuales en el ambiente. Participa en la generación en la generación y evolución de la escena determinando e informando sobre colisiones entre entidades dentro del ambiente. Además almacena y permite la generación de nuevos objetos virtuales, utilizando un lenguaje para la comunicación de acción y resultados.

- Comunidad de Agentes: Cada agente se encarga del comportamiento dinámico de las entidades activas o pasivas descritas para la escena e intenta cumplir con las metas asignadas por el usuario, de acuerdo a las habilidades y prioridades descritas para el agente, presenta una arquitectura propia, además de un control basado en algoritmos genéticos entre otros. La arquitectura de agentes esta diseñada para satisfacer, simulación de personalidad, emociones, administración de especificaciones de metas, noción de la forma grafica de si mismo y base de conocimiento. Permite además un ambiente de trabajo distribuido donde los usuarios puedan compartir agentes y habilidades para los agentes, el desarrollo de comportamientos permite una asignación dinámica de habilidades para los agentes, lenguaje de comunicación y protocolos de interacción.

- Núcleo GeDA-3D: Esta proyectado en esta visión de la arquitectura propuesta por el grupo, como un integrador y controlador de los módulos, y entre sus funciones define la manera en que los agentes interactúan para efectuar sus objetivos, es también el responsable de generar e integrar los módulos solucionando los problemas de heterogeneidad entre los mismo, proporciona un ambiente de desarrollo único para cada uno de los módulos que integran el sistema.

1.2 Descripción del Problema

El problema que se intenta resolver es el de la integración de los diferentes módulos que conforman la plataforma GeDA-3D. La integración de cada uno de los módulos mencionados y brevemente descritos en la sección anterior engloba requisitos inherentes y adquiridos. El propósito es desarrollar una plataforma "middleware" que permita el control de la evolución de la interacción entre entidades virtuales y el ambiente. Esta interacción permite a las entidades virtuales alcanzar los objetivos asignados por el usuario. El usuario generara los mundos virtuales utilizando un lenguaje semejante al natural que le permite describir la escena, las habilidades y las metas para cada una de las entidades activas y pasivas, así como las reglas que rigen a estas entidades [1.5].

1.3 Características y Requerimientos Para el Núcleo

En está sección se presentan los principales requerimientos en la integración de cada uno de los módulos del sistema GeDA-3.

1.3.1 Objetivos del Middleware

- Definir la manera en que los agentes interactúan para llevar acabo sus metas.
- Administrar la comunidad de agentes (Control de Agentes).
- Administración de especificación de metas y habilidades.
- Administración de Recursos.
- Capa de trasporte de mensajes.
- Generar y Administrar Conocimiento para los Agentes.

1.3.2 Requerimientos

- El sistema deberá administrar la interacción de las entidades virtuales conforme a las reglas naturales del mundo virtual generado.
- Implantar una base de datos conteniendo información fundamental de las entidades virtuales envueltas en el escenario.
- El sistema deberá asignar un agente para cada entidad virtual "avatares" (entidades representativas de forma gráfica).
- El sistema deberá administrar la comunicación entre ambiente, las metas y el contexto a los diferentes módulos.
- El sistema deberá realizar la recepción del módulo correspondiente y su envió de comandos a el modulo de representación gráfica.
- Administrar dinámicamente la creación de agentes y canales de comunicación por entidad virtual.
- Administrar eventos del tipo de colisión. Posponer o cancelar la ejecución de acciones en caso de colisiones o causas semejantes, controlar y regular el adecuado accionar de la evolución de la escena.
- El sistema será capaz de terminar la ejecución de un proceso ante la solicitud del usuario (eliminara cada una de la entidades virtuales y "avatares").

1.4 Organización de la Tesis

La organización empleada para el desarrollo de este documento es la siguiente:

- Capítulo 2 muestra un resumen sobre el estado del arte, y se estudia la capacidad de adaptación, para cumplir los requerimientos del sistema expuesto en la sección 1.3.2.
- Capítulo 3 presenta la propuesta y la arquitectura de la solución para el núcleo de GeDA-3D, se describe de forma concisa cada uno de los módulos y servicios que son parte de la propuesta y de la solución presentada.
- Capítulo 4 presenta un caso de estudio para el desarrollo de la solución propuesta, en la misma se precia la participación de los diversos servicios, generados para solventar los requerimientos del sistema.
- Capítulo 5 se expone de forma breve el tema de conclusiones y trabajo futuro en la investigación, resultado de la evaluación de la solución propuesta y de los retos propios del proyecto.

Capítulo 2 Estado del Arte

Resumen

Se analiza el panorama general de las diversas plataformas que puedan formar la base para construir la solución para algunos de los requerimientos para el núcleo del sistema GeDA-3D[1.1, 1.2], se estudia la capacidad de adaptación, para cumplir los requerimientos expuestos en la sección 1.3.2

2.1 Introducción

El proyecto de GeDA-3D puede ser abstraído como una proyecto que puede ser sostenido por el paradigma de la programación orientada a agentes[2.1], estos sistemas pueden emplear varias computadoras, las cuales se comunican empleando la infraestructura de red, en las cuales se distribuye una aplicación que demanda un rango de interacción para lograr una meta. Se investiga las posibilidades de emplear y explotar herramientas propias del campo de la Realidad Virtual (RV) e Inteligencia Artificial Distribuida (IAD), que son potencialmente adaptados para cumplir con los objetivos de este trabajo.

2.2 Marcos de Trabajo de Sistemas Multi-Agente (SMA)

A continuación se mostraran un informe de algunas de las diferentes plataformas de SMA que existen y cumplen con algunos de los requerimientos mostrados en la sección 1.3.2, los cuales forman parte de las necesidades de adaptación para el núcleo del sistema GeDA-3D. También, se muestra un breve análisis de cada una de las plataformas estudiadas, así como las oportunidades y retos que estas ofrecen hacia el cumplimiento de nuestros requerimientos.

2.2.1 AgentBuilder

AgentBuilder [2.1] es una suite de herramientas integradas para construir software de agentes inteligentes. Está desarrollada por Reticular Systems Inc. [2.1], y se basa en los modelos BDI [2.2, 2.3, 2.4] y Placa [2.5]. La herramienta destaca por la alta calidad y sus modelos, los cuales son modelos académicos bien conocidos. Este marco de trabajo es un producto comercial de código cerrado. Se dispone de una versión de evaluación, que solo permite utilizar los agentes tutoriales. Está disponible en tres versiones: AgentBuilder Lite, AgentBuilder Pro y AgentBuilder Enterprise [2.1]. También está disponible una versión académica, aunque todavía se encuentra en evolución.

Este sistema tiene la ventaja de manejar de manera transparente la movilidad, multiplataformas, sin embargo requiere alta especialización y no es posible realizar

extensiones y/o adaptaciones para añadirle nuevas características por lo tanto no cumple con los requerimientos de adaptación a GeDA-3D.

2.2.2 Aglets

Los Aglets [2.6] fueron desarrollados en los laboratorios de IBM [2.7] en Tokio [2.8]. Consisten en un grupo de clases e interfaces desarrolladas en el leguaje Java, estas permiten crear agentes móviles, un Aglet contiene un conjunto de estados que definen su forma de funcionar. Los principales eventos en la vida de un Aglet son: (Creación o Clonación, Liberación de recursos, Movilidad, Persistencia).

En este caso el sistema tiene la utilidad de poder ser extendido y los Aglets tiene completa independencia de ubicación, las desventajas es los agentes no son completamente distribuidos, solo pueden estar en un contexto local, no cumple con los requerimientos listados en la sección 1.3.2. Por lo tanto imposibilita su selección.

2.2.3 A-GLOBE

A-GLOBE [2.9] es una plataforma de agentes diseñada para probar escenarios experimentales. La plataforma proporciona los servicios tales como la comunicación, almacén, servicios de directorio y migración. Desgraciadamente es de código cerrado, no cumple con las especificación de FIPA, por lo anterior no será útil para los fines de esta investigación.

2.2.4 AMETAS

AMETAS [2.10] es la abreviatura para las siglas "Asynchronous MEssage Transfer Agent System", es un "middleware" para agentes móviles basada en aplicaciones distribuidas. AMETAS provee un ambiente de ejecución llamado "place" el cual debe ser instalado en todos los equipos a utilizar para este sistema. Cada "place" es un conjunto de servicios que le permiten a un agente la comunicación y migración. La licencia esta limitada solo a cuestiones académicas.

El soporte es muy precario. Se ha experimentado muy poco con la plataforma a gran escala. Ninguna consideración hacia el ambiente es realizada por el momento.

2.2.5 DIET Agents

DIET Agents [2.11] es una plataforma para sistemas multi-agentes, desarrollado en Java como parte del proyecto DIET, fue realizada bajo el diseño ascendente, se pueden realizar de forma rápida prototipos de aplicaciones punto a punto. Es de arquitectura abierta.

No se hace un manejo de arquitectura para la administración de los agentes, tampoco se consideran las metas, esta enfocado a el manejo solo de agentes y no se considera el ambiente en las simulaciones.

2.2.6 JACK

El marco de trabajo JACK [2.12, 2.13] es un entorno comercial basado en JAVA para la construcción, ejecución e integración de SMA sobre un enfoque basado en componentes. Fue creado por Agent Oriented Software Pty. Ltd., Está basada en el modelo BDI ("Belief-Desire-Intention") [2.4] y dMARS [2.14] desarrollado en el Instituto Australiano de Inteligencia Artificial (AAII). JACK está enfocado principalmente en la etapa de desarrollo. Los pasos de diseño y análisis se mencionan únicamente en [2.12].

Esta herramienta asume que se ha proporcionado la definición de funcionalidad de los agentes, y que se ha elegido el modelo BDI [2.4]. Tiene una carencia de métodos de diseño, es necesario conocer profundamente su modelo BDI. No cumple con los requerimientos listados en 1.2.1.

2.2.7 JADE

JADE [2.15, 2.16] ("Java Agent Development Framework") es el middleware desarrollado por TILAB [2.17] (Telecom Italia LAB) para el desarrollo de sistemas de aplicaciones distribuidas de SMA basadas en la comunicación punto a punto. Tanto la inteligencia, la iniciativa, la información, los recursos y el control pueden ser completamente distribuidos en terminales móviles. JADE es una tecnología que posibilita un middleware para desarrollo y ejecución de aplicaciones punto a punto, las cuales son basadas en el paradigma de agentes.

JADE es desarrollado totalmente en Java y es de código abierto, incluye un servicio de nombres que asegura que cada agente tenga un nombre único y un servicio de páginas amarillas, trabaja de forma distribuida y utiliza la tecnología punto a punto.

La estructura de mensajes cumple con la definición de lenguaje FIPA-ACL [2.18]. Jade es ampliamente utilizado por la comunidad académica y varias compañías lo utilizan para desarrollar sus proyectos internos, incluyendo Telecom Italia [2.17], Whitestein Technologies AG [2.19] y Rockwell Automation [3.20].

JADE es una herramienta para el desarrollo de SMA genéricos que cumple con todos los estándares de FIPA [2.21], cumple con la mayoría de los requerimientos de GeDA-3D listado en sección 1.3.2 y puede ser adaptado para cumplir con los requerimientos 1.3.2-5 y 1.3.2-7. Actualmente existe una extensión de esta plataforma denominada JADEX [2.22] es una extensión BDI [2.23, 2.24] para la plataforma JADE, creada por el grupo de sistemas distribuidos de la universidad de Hamburgo en Alemania con esta extensión se han desarrollado múltiples proyectos[2.25].

2.2.8 MadKit

MadKit [2.26, 2.27] es una plataforma Java construida sobre un modelo organizacional. Está desarrollada por Olivier Gutknecht y Jacques Ferber en el LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier), un laboratorio público de investigación en Francia. MadKit es principalmente un motor de ejecución SMA, utilizando un micro-núcleo de agente. El modelo organizacional subyacente se llama Aalaadin [2.27, 2.28]. El modelo representado en MadKit está formado por agentes heterogéneos agrupados en algún conjunto, el SMA como un todo es la unión de estos conjuntos de agentes. Así también, cada agente en los conjuntos tiene una o más funciones, además de su identificación.

En resumen MadKit está enfocado en aspectos de cooperación y organización más que en la capacidad y estructura de los agentes, esta basado en un modelo de eventos. Aun así con MadKit no se asocia ningún método de análisis, no se proporciona ningún modelo de agente, por lo tanto tiene que ser implementado.

2.2.9 RETSINA

RETSINA ("Reusable Environment for Task-Structured Intelligent Network Agents") [2.29, 2.30], es una bien conocida infraestructura de SMA, RETSINA es una infraestructura abierta de SMA que soporta comunidades de agentes heterogéneos.

El ACL utilizado en Retsina es KQML ("Knowledge Query and Manipulation Language") [2.31] y un lenguaje de protocolo. Retsina provee un servicio para monitorear el desempeño de simulaciones. Asimismo provee una ontología derivada de la "Wordnet Ontology" [2.33]. Por otra parte Retsina utiliza un lenguaje llamado LARKS ("Language for Advertisement and Request for Knowledge Sharing") [2.32] para compartir el conocimiento entre agentes. La Retsina-OOA es un ínter-operador superior de la arquitectura SMA de Retsina, la cual estable un puente entre esta con la plataforma OOA (Open Agent Architecture) [2.34].

Sin embargo no se hace ninguna consideración en el modelo de representación del conocimiento en un aspecto de conocimiento privado de cada uno de los agente, por lo tanto se requiere de un trabajo extra para instanciar este marco de trabajo, y no se tiene soporte para movilidad y tiene carencias en la seguridad.

2.2.10 SAGE

El proyecto SAGE [2.39] tiene el objetivo de un desarrollo una plataforma distribuida descentralizada tolérate a fallas, escalable y que cumple con las nuevas especificaciones de FIPA, pero demanda muchos recursos y no tiene un soporte adecuando, no presenta manejo alguno de ambientes, ni metas.

2.2.11 Zeus

Zeus [2.35, 2.36] es un entorno integrado para construcción rápida de aplicaciones basadas en agentes colaboradores, Está desarrollado por el programa de Investigación de Agentes del British Telecom Intelligent System Research Laboratory. "La metodología de creación de agentes es vital para usar el conjunto de herramientas de Zeus" [2.36]. El modelo de agente de Zeus limita el diseño a los agentes colaborativos orientados a tareas y conducidos por objetivos. Esta etapa requiere principalmente habilidades sobre todo en el diseño, pero también es obvio que existen carencias en el formalismo. Las principales actividades involucradas en el desarrollo de los agentes son auxiliadas por medio de herramientas gráficas de software. Algunas de estas actividades son: creación de ontología, creación de agente, configuración de agente, implementación de agente.

Estas actividades requieren un gran conocimiento de las herramientas, para agregar una nueva funcionalidad requiere volver al código. Sin embargo, aún cuando

Zeus es modular, se soporta un único modelo de agente, lo que limita la gama de posibles diseños de SMA. Por otra parte Zeus es difícil de dominar, tienen carencias de formalismo, además no se proporciona ningún soporte oficial.

2.3 Trabajos Relacionados

Existen actualmente muchos trabajos que están relacionados con el proyecto GeDA-3D, en la presente sección se describen algunos de ellos. Recientemente este tema que es el de desarrollo de aplicaciones con agentes en ambientes 3D, a tomado un nuevo auge, tenemos ahora la explosión de resultados de algunos de ellos, los cuales presentaremos a continuación.

2.3.1 Proyecto INEVAI 3D

El Proyecto INEVAI 3D [2.40, 2.41] (Integración de Escenarios Virtuales con Agentes Inteligentes 3D) pretende definir un marco global y unificado de los sistemas actuales y futuros en entornos virtuales con agentes tridimensionales inteligentes. Ellos exponen objetivos amplios y ambiciosos, se puede simplificar en los siguientes cuatro puntos:

a) Integración Unificada de Escenarios Virtuales (Web, GSM, UMTS, Chat, TV, 3D, etc.).
b) Simulación de Humanóidees Inteligentes (Agentes).
c) Análisis de Movimiento Humano mediante sensores de fuerza.
d) Nuevos Interfaces multimodales y su aplicación a entornos domóticos con Agentes 3D.

El proyecto desgraciadamente esta en curso y no presenta muchos resultados aun, sin embargo en un futuro podría producir aportes para la comunidad. Actualmente están la investigación y desarrollo en curso, sin que se logren objetivos concretos aun.

2.3.2 Proyecto DEAPspace

EL proyecto DEAPspace [2.42] esta basado en proporcionar soporte a la computación distribuida punto a punto entre dispositivos. Un aporte de este proyecto es la definición de un protocolo de descubrimiento y anuncio de servicios llamado DEAPspace Algoritmo [2.42], por medio de este algoritmo se detectan dispositivos y se comparte información de los servicios que están disponibles en la red y detectar cuando

un dispositivo deja de estar disponible. Actualmente la investigación que representa este proyecto no esta activa, pero las expectativas para el proyecto son mejorar los problemas de seguridad y configuración entre otros.

2.3.3 Worlds Studio

Worlds Studio [2.43] se proyecta como herramientas y métodos innovadores de computación para la producción de mundos virtuales 3D, La meta principal de este proyecto es crear un sistema o conjunto de herramientas para generar ambientes multi-usuario para mundos virtuales en 3D, de una forma rápida y fácil, incluyendo paisaje e interactividad. La plataforma final del desarrollo del proyecto de Worlds Studio, permitirá a un marco unificado, la creación 3D y la adición de la interactividad. Todo esto para proporcionar una solución para generar ambientes 3D virtuales profesionales y compartidos. El proyecto esta dirigido a construir una plataforma integrada para la producción de mundos virtuales 3D para la comunidad, de acuerdo con sus experiencias este proyecto constituye un paquete de software que proporcione la mejor solución para tratar las necesidades de desarrollos virtuales 3D de la comunidad en Europa. Pero el proyecto no ha sido liberado hasta el día de hoy.

2.3.4 AréVi: Atelier de Réalité Virtuelle

Fue desarrollado en el centro europeo de realidad virtual en Francia [2.44], el objetivo del proyecto es definir y desarrollar métodos y las herramientas interactivas para prototipos de colaboración y cooperación en ambientes virtuales distribuidos. Esta biblioteca esta diseñada para simular agentes autónomos en un ambiente 3D [2.45, 2.46]. Este API proporciona los servicios para sistemas multi gráficos 3D (ver Figura 2.1) en SMA.

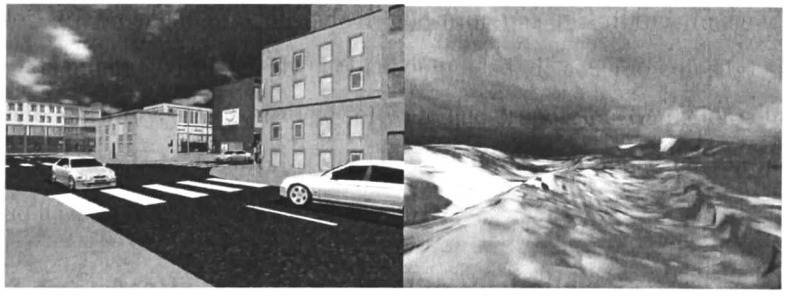


Figura 2.1 Dos escenarios posibles en AréVi

La plataforma presenta introspección / reflexión. En lo referente a la investigación este proyecto se basa específicamente sobre el modelado del comportamiento de agentes, la comunicación entre las entidades, los ambientes virtuales y de las plataformas de los ambientes virtuales distribuidos.

La meta es poder describir las entidades autónomas, capaces de razonar en las acciones que pueden hacer. El modelar del comportamiento: Se procura estudiar y modelar más específicamente los comportamientos humanos bajo el ángulo de la psicología y de la neuropsicología experimental.

En lo que respecta a la comunicación entre las entidades: La meta es permitir la comunicación masiva y a usuarios utilizando el mismo lenguaje de comunicación. Por otra parte es de desatar que este trabajo resulta por demás interesante por la relación directa con nuestro proyecto.

2.4 Middleware

El concepto de "Middleware" se desarrollo en los 90s como un apoyo para solucionar los problemas de heterogeneidad en los sistemas distribuidos [2.47], capaces de construir sistemas distribuidos eficaces e integrados por diversos componentes. El Middleware nos excluye de complejidad y heterogeneidad de las redes de comunicaciones, sistemas operativos y lenguajes de programación subyacentes [2.49]. En función del problema por resolver serán favorables el empleo de algunos de los diversos tipos de servicios de "middleware"

Middleware es absolutamente un término general que cubre una variedad de software. Puede ser separado en diversas categorías, y éstos son los tipos más comunes:

2.4.1 Middleware Orientado a Transacciones

Apoya las transacciones entre los componentes de un sistema distribuidos. El middleware orientado a transacciones se asegura de que los procesos distribuidos de las transacciones cumpla con las siguientes propiedades, que todas las transacciones sea: atómicas, consistentes, aisladas y durables [2.48].

2.4.2 Middleware Orientado a Componentes

Un componente es una unidad de composición de aplicaciones Software. Un middleware orientado a componentes es una disposición de los diversos componentes,

que se pueden ser constituidos en tiempo de ejecución. Un ejemplo de este es .NET de Microsoft [2.50].

2.4.3 Middleware Orientado a Servicios

Un middleware orientado servicios es básicamente una recopilación de servicios [2.51]. Un servicio realiza algunas funcionalidades y tiene las características siguientes:

1. La interfaz a el servicio x es independiente de la plataforma.
2. El servicio puede localizarse e invocado dinámicamente y es autónomo.

4.3.1 Middleware Orientado a Procesos

Es una extensión de la interfaz "Remote Procedure Call" (RPC) y ofrece unas invocaciones transparentes del procedimiento sobre una red [2.52]. El RPC fue ideado por Sun Microsystems a inicio de los años ochentas.

2.4.5 Middleware Orientado a Mensajes

Esta diseñado para ofrecer servicios de mensajería asíncrona entre aplicaciones [2.52]. También permite distintos tipos de comunicación como la basada en el P2P o como la basada en productores consumidores. En P2P los mensajes son enviados y recibidos de un destino en común establecido por los componentes. En el otro modelo los componentes se suscriben para publicar o consumir mensajes del mismo. Para la comunicación e intercambio de información entre componentes se define un protocolo basado en mensajes, flexible e independiente de la plataforma.

2.4.6 Middleware Orientado a Objetos

Tiene las base del objeto, como son herencia y polimorfismo, la comunicación entre los objetos puede ser síncrona/asíncrona. El estándar utilizado es "Common Object Request Broker Arquitecture" (CORBA) [2.53], desarrollada por "Object Management Group" (OMG) [2.53], y es el middleware distribuido más amplio orientado a objetos. Existen diversas implementaciones del estándar como: Orbix, Inprise VisiBroker y JavaIDL.

2.5 Resumen

El auge en el desarrollo de plataformas multi-agente nos ofrece herramientas muy útiles e interesantes, como lo son JADE [2.15, 2.16], JACK [2.12, 2.13], MadKit

[2.26, 2.27] y Zeus [2.35, 2.36]. Actualmente existen mucha herramientas sobre las cuales se pueden facilitar el desarrollar SMA, algunas de estas están centradas en coordinación y comunicación [2.37], otras en movilidad [2.8], otras son difíciles de aprender a utilizar [2.35] y muchas ofrecen participar en el enriquecimiento del proyecto como son entre otras Jade y MadKit. Aun con todos los esfuerzos realizados existen carencias para facilitar la creación y/o simulación mediante SMA. Hay mucho trabajo por hacer para tener herramientas capaces de facilitar el desarrollo integral de SMA. Por otro lado los trabajos relacionados con GeDA-3D, que se exponen en la sección 2.3, se pueden clasificar como recientes. Dichos trabajos son algunos muy ambicioso como Proyecto INEVAI 3D [2.40, 2.41], otros han tenido un receso en su evolución como DEAPspace [2.42], pero todos tiene el objetivo de facilitar la simulación para los usuarios mediante el empleo de técnicas de la inteligencia artificial distribuida y/o realidad virtual, estos trabajos demuestran el creciente interés por la comunidad hacia el tema.

2.6 Conclusiones

La evolución de los sistemas distribuidos y el interés de la comunidad, nos han legado una amplísima variedad de posibilidades para hacer frente y solucionar los requerimientos del sistema GeDA-3D mencionados en la sección 1.3.2, de las cuales se ha realizado una evaluación para determinar la factibilidad y conveniencia de su adaptación para cumplir con los requerimientos que demanda el proyecto. Con base en todo lo anterior y considerando todas las opciones se presenta en el siguiente capitulo una propuesta y arquitectura para el núcleo de GeDA-3D.

Capítulo 3 Solución Propuesta

Resumen

Se presenta la solución propuesta para el desarrollo de la arquitectura e implementación para el núcleo de GeDA-3D, se describen los servicios demandados y generados para cada uno de los módulos.

3.1 Introducción

GeDA-3D [1.1, 4.2] es un proyecto ambicioso, el cual tiene como objetivo desarrollar una herramienta que utilice la inteligencia artificial distribuida para proporcionar a usuarios inexpertos la facilidad de describir mediante un lenguaje semejante al natural, diversas ambientes en los cuales se desarrollaran las simulaciones basadas en comportamientos, dentro de los tipos principales de simulaciones tenemos:

1. *Aplicaciones semi-interactivas*: Narrador Virtual: Permite a un usuario inexperto obtener una representación de un guión por medio de escenas en 3D.
2. *Aplicaciones no interactivas*: En este tipo de simulaciones el usuario únicamente describe: comportamientos y/o competencias y/o personalidades y/o emociones de cada entidad, describe un ambiente y permite su evolución completamente autónoma. Existen muchas aplicaciones interesantes para este tipo simulaciones tales como comportamientos animales, de colonias de bacterias, de fenómenos tales como el fuego, etc.
3. *Aplicaciones interactivas:* Este tipo de aplicaciones proporcionan el potencial al usuario para modificar en tiempo de ejecución cualquier característica en la simulación. Algunos de los ejemplos que podemos mencionar para este tipo de aplicaciones son la simulación de equipos tales como reactores, los cuales pueden dar una experiencia aproximada al usuario en el empleo de este tipo de equipo.

El objetivo final es que GeDA-3D proporcione el soporte necesario a cada uno de estos tipos de aplicaciones.

3.2 Planteamiento del Problema

Generar un núcleo que permita la integración de los módulos necesarios para implementar los tres tipos de aplicación descritos anteriormente. Como se pudo mostró en el capitulo referente al estado del arte, existen una amplia gama de posibilidades para integración, entre ellas: la modificación de plataformas relacionadas de arquitectura abierta, realizar la implantación de un "middleware" apropiado.

3.3 Solución Propuesta

La solución propuesta es desarrollar un "middleware" orientado a mensajes [3.1], el cual proporcione servicios establecidos en un documento de análisis de requerimientos. Esta decisión se tomo teniendo en cuenta: que no existe entre las opciones evaluadas en la pasada sección una herramienta que resuelva los requerimientos demandados por el proyecto y las que se aproximan debe ser reestructuradas y violarían en gran parte la autonomía de los diversos módulos. Los servicios que debe proporcionar el núcleo comprenden grosso modo: servicio de nombre, servicio de direcciones, y servicios especializados para algunos módulos como son el servicios de administración de agentes, servicio de control y evolución de escena, son consecuencia directa de las necesidades de adaptación y por otra parte referentes funcionales necesarios para cumplir con los objetivos del proyecto. A continuación se describe la arquitectura de la solución propuesta.

3.4 Arquitectura

En esta sección se presenta la arquitectura que satisface los requerimientos, para y características descritos en las secciones 1.3.2 y 1.3.2. En la Figura 3.1, se puede ver una visión general de la solución completa, la cual muestra una abstracción de la arquitectura propuesta, para la integración de cada uno de los diferentes módulos que integran la plataforma de GeDA-3D.

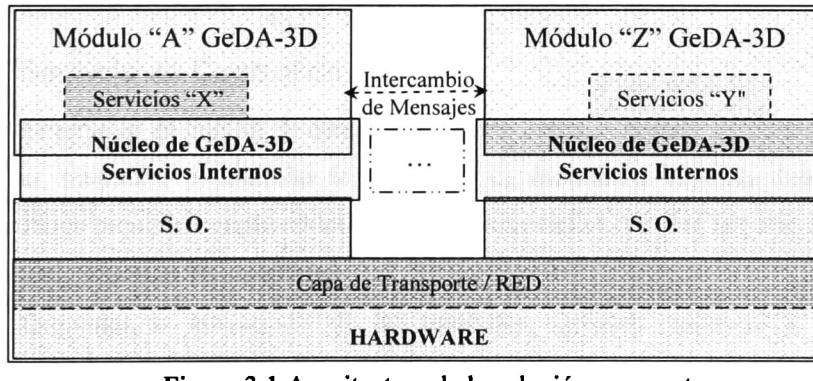


Figura 3.1 Arquitectura de la solución propuesta

La Figura 3.1 muestra que la comunicación entre los módulos o aplicaciones se realiza por medio de paso de mensajes asíncronos, de esta forma cualquier módulo que

se conecte al sistema GeDA-3D, puede enviar un mensaje sin esperar la respuesta. Los mensajes son tratados como unidades auto-contenidas ya que ellos contienen los datos necesarios para que puedan ser procesados. El "middleware" asegura que los mensajes son distribuidos convenientemente para cada uno de los componentes participantes del sistema. En lo relacionado a la comunicación se permite comunicación punto a punto y comunicación multipunto. Más delante son tratados a detalle cada uno de los servicios principales en los cuales el núcleo tiene una participación directa o son solventados por el mismo.

3.5 Servicios

Los servicios aportan diversas herramientas útiles para cumplir los objetivos del núcleo de GeDA-3D, a continuación en la presente sección se describen los principales servicios desarrollados.

3.5.1 Servicio de Trasporte de Mensajes

El servicio de trasporte de mensajes es un servicio esencial, permite el intercambio de información, logrando una facilidad de comunicación entre los módulos. Los mensajes son auto-contenidos, el formato de alto nivel permite especificación precisa de contenidos de información y mensajes trasmitidos entre los elementos, con lo cual este servicio nos permite tener una invasión mínima en el ámbito de código y en el sentido operativo, lo cual se garantiza la menor violación a la autonomía de cada uno de los componentes del sistema, facilitando además la apertura del proyecto.

3.5.2 Servicio de Control de Escena

El servicio de control de escena tiene como objetivo normar la interacción entre todas las entidades, y controlar la evolución de escenarios virtuales deterministas, dentro de los requisitos establecidos por el proyecto GeDA-3D (ver sección 1.3.2) para este servicio tenemos los siguientes:

- Orquestar la interacción de las entidades virtuales conforme a las reglas estipuladas para el mundo modelado.
- Validar y controlar los efectos de interacción entre entidades.
- Recepción periódica de acciones venida desde los agentes activos.

- Validar la ejecución de todas la acciones recibidas, el habilitará retardos o cancelara la ejecución de una simple acción o una secuencia de acciones.
- Contendrá una base de datos conteniendo información fundamental de las entidades virtuales envueltas en el escenario.

Un aspecto a desatacar de este servicio es el ciclo de acciones y reacciones. Una vista estructural de cómo se maneja el control de la evolución de la escena por este servicio es la mostrada en la Figura 3.2.

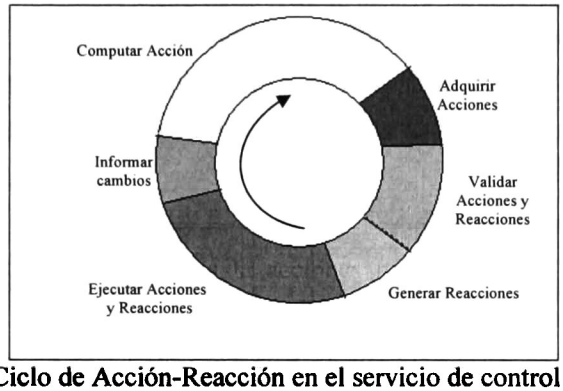


Figura 3.2 Ciclo de Acción-Reacción en el servicio de control de escena.

De la Figura se observar que el modelo de acción-reacción es equiparable con el modelo de ejecución-reacción [3.1]. Algunas de las discrepancias son debidas a la realización de mejoras y adaptaciones para cumplir con nuestras metas en el servicio. Con lo cual tenemos que el ciclo inicia dando información del estado actual del ambiente a cada uno de las entidades activas, se continua con una espera para que los diferentes algoritmos en los diferentes agentes, si algún algoritmo dentro del tiempo "Computar Acción" no logro generar un acción esta será ignorada, las acciones que se recolectaron dentro de ese tiempo de "Adquirir Acciones" será evaluadas, por medio del contexto modelado para el mundo virtual en ejecución, las acciones validas como correctas será realizadas y se provocan las reacciones como consecuencia de su accionar. En el mismo tiempo se llevan acabo en el módulo que despliega gráficamente los resultados para el usuario, y se informa a las entidades de los cambios generados, luego el ciclo se reinicia. Es así como se solventan los requerimientos y se proporciona un control dinámico de la evolución de la escena en el ambiente virtual modelado.

El servicio ofrece además la posibilidad de modificar el contexto en tiempo de ejecución, para lograr diversas trasformaciones en el comportamiento y evaluación de la escena en tiempo de ejecución.

3.5.3 Modelado del Servicio de Contexto usando UML y Redes de Petri

Este servicio es una extensión en la participación del uno de los módulos con el núcleo el cual se realiza basado en ontologías por parte del módulo, este mismo determina el contexto y las acciones permitidas en este así como las precondiciones y las poscondiciones necesarias para que una acción se considerada valida en por el control de escena, es en si una estructura bajo la cual se puede determinar reglas y consecuencias de estas. Una visualización del esquema UML del contexto se muestra en la Figura 3.3.

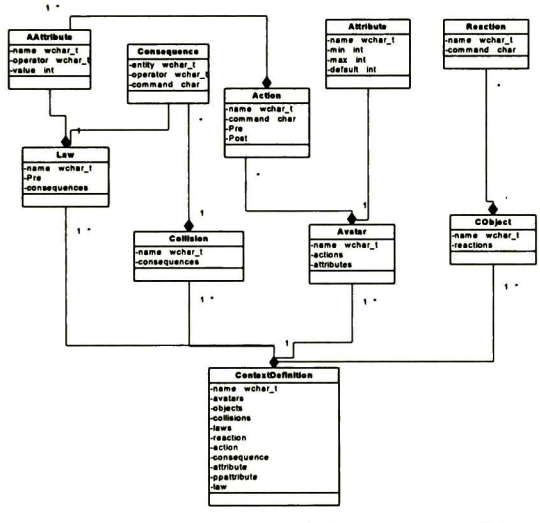


Figure 3.3 Esquema UML de la estructura del contexto

Otros de los aspectos considerados en la definición del contexto son las colisiones, las entidades pasivas y las leyes que gobernaran la evolución de la escena, otro de los aspectos importante de este tipo de estructura es la facilidad para el manejo por parte del servicio de control de escena descrito anteriormente.

El esquema UML del servicio de contexto nos presenta un panorama general de cómo se estructuro la representación de contexto en el sistema. A continuación en la Figura 3.4 se muestra y describe brevemente un modelo formal por medio de RPC.

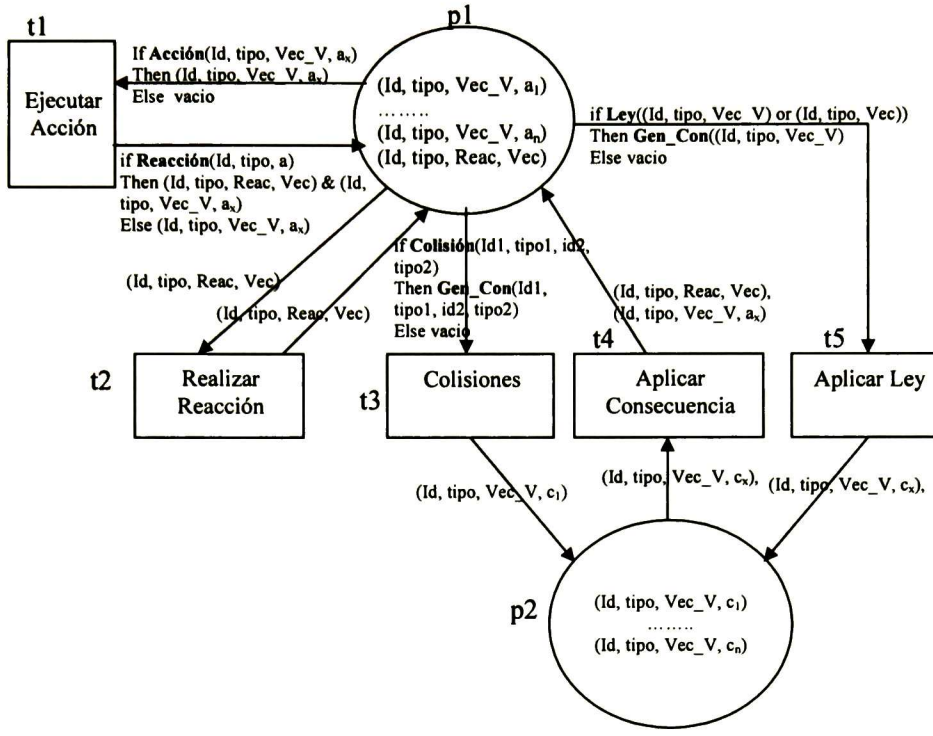


Figura 3.4 Simple representación del contexto por medio de RPC.

Dentro del modelo presentado en la Figura 3.4 el lugar "p1" representa las condiciones iniciales descritas por el usuario por medio del marcado inicial del sistema y contenedor de entidades actuantes en el sistema. En dicho lugar se destacan dos conjuntos de arreglos de colores, dentro del primer paquete de colores tenemos: que color "id" representa el identificador de la entidad; "tipo" que representa el tipo de entidad, un color denominado "Vec_V" determina el conjunto de características de la entidad, y por ultimo en el paquete de colores tenemos el color identificado como "a_x" que define la acción solicitada por la entidad. El otro paquete de colores define lo siguiente: primero el color identificado con "id" determina el identificador pero en este caso de un objeto inanimado; el color "tipo" también nos representa el tipo de objeto, así mismo el identificador "Vec_V" se relaciona con un vector de características del objeto, por ultimo, el color identificado con la etiqueta de "Reac" determina la reacción activa del objeto inanimado. Por otra parte para el lugar nominado "p2" tenemos la siguiente representación para cada uno de los elementos de los paquetes de colores contenidos en esté: el primer elemento del paquete de colores es "id" que simboliza la

variable que representa el identificador de la entidad para el sistema, "tipo" que identifica el tipo de la entidad, "Vec_C" es el vector dinámico de características de la entidad, y por ultimo tenemos el color "C_x" que determina el tipo de consecuencia que se aplica a la entidad.

Para el caso de las transiciones podemos hablar de las funciones lineales de colores, primero para la función del arco (p1, t1) se puede apreciar un método denominado "Acción(Id, tipo, Vec_V, ax)", el cual verifica si la acción solicitada por la entidad es valida de acuerdo a los parámetros y características de la misma, en caso de que la acción no sea valida se deshecha o si es valida permite que esta sea ejecutada, y la función del arco (t1, p1) tenemos las consecuencias de la ejecución de tal acción para lo cual se ejecuta un método nombrado "Reacción(Id, tipo, a)" la cual evalúa si tal acción genera una reacción o no y según el caso enviara la marca o marcas a el lugar identificado como p1, para el la función del arco (p1, t2) se habilita cuando ocurre paquete de colores de la estructura "(Id, tipo, Reac, Vec)" entonces el sistema realiza la reacción indicada, ahora para la función del arco (t2, p2) tenemos que se retorna la marca a el lugar p1 con la modificación producida por la ejecución de la reacción, continuando con las funciones de los arcos tenemos ahora la función para el arco de (p1, t3) el método "Colisión(Id1, tipo1, id2, tipo2)" verifica si es existen colisiones entre las diferentes entidades, y si existen son generadas la estructura de colores "(Id, tipo, Vec_V, c1)" mediante el método "Gen_Con(Id1, tipo1, id2, tipo2)" y estas marcas son colocadas en el lugar denominado "p2" que actúa como una contenedor de las entidades a las que se les aplicaran las consecuencias determinadas tanto por la transiciones "t3" y "t5", en el caso de la función del arco (t3, p2) al igual que el arco (t5, p2), coloca las entidades participantes en la colisión dentro del contenedor ((Id, tipo, Vec_V, c1)), junto con las consecuencias producidas por dicho evento, la función del arco (t4, p1) es la de retornar el paquete de colores después de que son aplicadas las consecuencias sobre las entidades, por último tenemos la función del arco (p1, t5) en la cual el método "Ley(Id, tipo, Vec_V)" determina si una ley es aplicable para la entidad y el método "Gen_Con((Id, tipo, Vec_V)" genera las consecuencias para la entidad las cuales son marcas colocadas en el lugar "p2", y por ultimo estas marcas contenidas en "p2" son transferidas por los arcos (p2, t4) y (t4, p1) por medio de la transición "t4", que ejecuta las consecuencias sobre las entidades y realiza las modificaciones en sus vectores de características.

3.5.5 Servicio de Conversión de ACL/LIA-3D

Este servicio trasforma comandos de formato ACL a su correspondiente en LIA-3D de forma practica y segura, la solución presentada para este requerimiento establece el análisis para determinar que acciones son admitidas por LIA-3D, esto es determinado por la base de datos de acciones disponibles por avatar, esta base de datos esta dentro del sistema AVE-3D.

```
- <context name="predator-prey">
  - <avatars>
    - <avatar name="Predator">
      - <attributes>
          <attribute name="Energy" min="0" max="200" default="200" />
        </attributes>
      - <actions>
        - <action name="Default" command="STATIC">
          - <pre>
            - <attributes>
                <attribute />
              </attributes>
            </pre>
          - <post>
            - <attributes>
                <attribute />
              </attributes>
            </post>
          </action>
        - <action name="DANCE" command="DANCE">
          - <pre>
            - <attributes>
                <attribute />
              </attributes>
            </pre>
          - <post>
            - <attributes>
                <attribute />
              </attributes>
            </post>
          </action>
```

Figura 3.5 Representación en formato XML

De esta forma se pueden ver por un medio contextual cuales acciones pueden realizar los agentes, y que consecuencias tienen estas acciones. También se determinan los comandos equiparables de las acciones del agente a las acciones de los avatares.

3.5.6 Servicio de Administración de Agentes

El servicio de administración de agentes es uno de los servicios medulares del sistema apoya al control de escena en la administración de recursos, este registra las diversas características y habilidades que los agentes tiene disponibles, es responsable de instanciar los agentes y apoya en la vinculación de los agentes con los avatares. El administrador de agentes asigna coordenadas y metas conjuntas e individuales, a cada uno de los agentes. También se encantar de eliminarlos y/o ponerlos en marcha,

asignarles nombres e identificadores a cada agente en ejecución. Una vez realizada esta actividad los agentes se comunican de forma autónoma al servicio de control de escena el cual norma las actividades y el desarrollo de la escena.

3.5.7 Servicio de Representación de Conocimiento

La representación del conocimiento cubre aspectos como son la toma de decisiones, el razonamiento sobre evidencias, el razonamiento basado en casos, el razonamiento por analogía e inducción. La representación del conocimiento en el sistema es por medio de lógica de predicados, con la cual se cumple el objetivo para este servicio, la simbología se muestra a continuación:

Conectivos preposicionales: $\neg, \wedge, \vee, \Leftrightarrow, \Rightarrow$

Cuantificadores: $\forall$ universal y $\exists$ existencial

Las proposiciones son representados mediante símbolos (P, Q, R,...) y estas son evaluadas a verdaderas o falsas.

La construcción de las preposiciones puede hacer uso de conectivos lógicos como la conjunción "$\wedge$", la disyunción "$\vee$" y la implicación "$\Rightarrow$" de esta manera se pueden construir proposiciones compuestas del tipo $((\mathbf{P} \wedge \mathbf{Q}) \Rightarrow \mathbf{R})$ que se forma a partir de las proposiciones **P**, **Q** y como consecuente la proposición **R**.

La generación de inferencia es realizada mediante encadenamiento hacia adelante. El procesamiento es realizado de la siguiente forma:

1. Se dispone de todas las reglas de base para la maquina de inferencia, así como de todos los hechos contenidos en la base de conocimiento.
2. Se determina en base a los hechos, cual de las reglas esta habilitada de acuerdo a sus antecedentes.
3. Se dispara la regla con más antecedentes y que esta habilita, se cargan las consecuencias de dicha regla en la base de conocimiento.
4. Se continúa con el paso 2 hasta que ya no exista regla alguna por ser habilitada.

De esta forma los agentes pueden enriquecer su conocimiento. El conocimiento lo dividimos en los dos tipos siguientes en función de su accesibilidad.

1. Visto y accedido por cualquier agente del mundo virtual modelado,

2. Solo accesible a un grupo del universo de agentes en el mundo virtual modelado.

3.5.8 Servicio de Administración de Recursos

Cada uno de los módulos requiere y demanda diversos recursos locales y remotos, el registro de los recursos es una forma en la cual se pueden publicar los recursos disponibles en la plataforma, muchos de los recursos solicitados en la red pueden ser parte de los recursos locales de un nodo del sistema, el cual al recibir la solicitud envía de regreso el recurso solicitado, cuando un recurso no es localizado, el administrador de recursos genera una excepción indicando que no se dispone del recurso solicitado.

Muchos de los recursos son archivos o imágenes que son solicitadas por el nodo de renderización, el cual los emplea para realizar representaciones de las criaturas virtuales u objetos virtuales dentro del mundo virtual modelado. También se comparte habilidades y en general cualquiera de este tipo de recurso que los nodos puedan requerir.

4.3 Conclusiones

Los diversos servicios son resultado de la necesidad de adaptación de los diferentes módulos que componen el sistema denominado GeDA-3D, el núcleo genera estos servicios para solventar las discrepancias y necesidades de los módulos dentro de la plataforma. Algunos de los servicios como son el "Administrador de Agentes" son necesarios para determinar los recursos con respecto a las habilidades y agentes que existen dentro de la plataforma. Otro semejante a este es el "Administrador de Recursos" el cual es encargado de administrar los recursos, los cuales pueden ser archivos que representan objetos virtuales, mundos o escenarios virtuales que algunos de los componentes del sistema necesitan para su correcta operación. El ciclo de ejecución propone una manera práctica para controlar la evolución del escenario virtual. Para terminar podemos también concluir que el servicio de representación de conocimiento basado en lógica proposicional puede ser empleado para modelar formalmente los elementos de un escenario por ejemplo objetos, funciones e interrelaciones de las entidades en el mundo virtual.

La solución presentada es una solución abierta y permite la evolución individual de los diversos módulos, ya que es mínimamente invasiva.

Capítulo 4 Casos de Estudio

Resumen

En esta sección se presentan los casos de estudios, los cuales fueron desarrollados para dos visores gráficos distintos, los cuales aportan mundos virtuales diferentes.

4.1 Introducción

GeDA-3D [1.1, 1.2, 1.3] proporciona herramientas que facilitan el desarrollo de aplicaciones distribuidas, estas se basan en paradigmas de la inteligencia artificial distribuida, para proporcionar a usuarios inexpertos la facilidad de desarrollar simulaciones por medio de un lenguaje semejante al natural, es por este lenguaje que se determinan las condiciones iniciales y las características de la simulación que el usuario desea desarrollar. Las herramientas que se emplearon para el desplegado de resultados en formato gráficos son dos. La primera es AVE-3D [4.1] que es una herramienta producto de una tesis del equipo. La segunda herramienta esta basada en OPEGL [4.2] y es producto de investigación de otro trabajo del grupo [4.3, 4.4].

4.2 Batalla de Ranas Utilizando Agentes Emocionales

Este caso de estudio fue generado para determinar el alcance y versatilidad de la solución propuesta en este caso de estudio se muestran como las emociones afectan los comportamientos y como es que se puede controlar la generación de entidades en el ambiente y la evolución de los agentes y objetos dentro del ambiente [4.6].

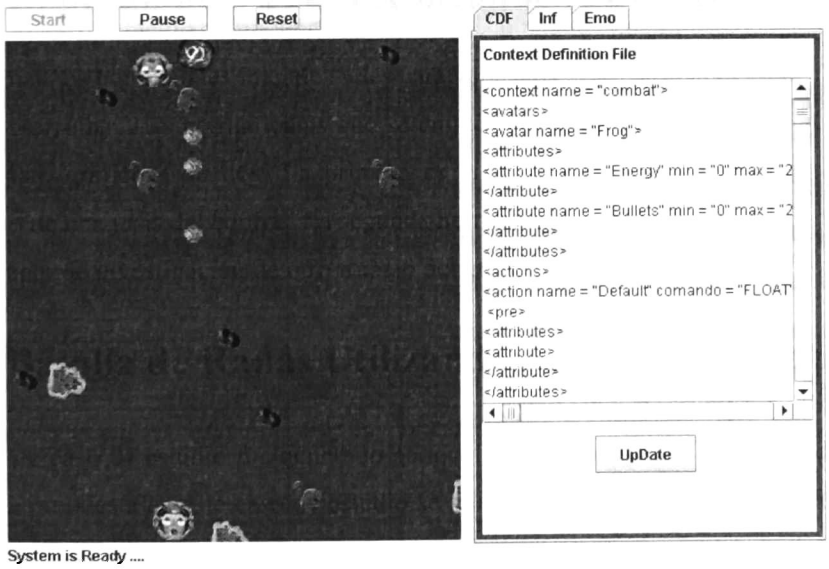


Figura 4.1 Agentes emocionales y servicios básicos.

Con la generación de este caso de estudio se prueba que tanto la definición de contexto, como el uso de ontologías para la definición del contexto son muy útiles y pueden ser empleadas con muy buenos resultados. El empleo de reglas y leyes que gobernarían este ambiente nos proporciona las bases para el control de la evolución de la escena en el mundo virtual modelado.

4.2.1 Resultados

Los resultados obtenidos fueron los siguientes:

- Las entidades son regidas de forma dinámica por medio de reglas.
- Se incluye la influencia emocional en cada uno de los agentes participantes.
- El manejo de reacciones y acciones es por medio de un ciclo bien definido.
- Se controla el ciclo de vida de los agentes de acuerdo a reglas o leyes generadas por los usuarios.
- Se logra una representación clara y concisa de las características de las entidades en el ambiente.
- Se permite la evaluación de comportamientos afectados por las emociones.

4.3 Depredador - Presa Aplicando AVE-3D

El presente caso de estudio requirió de servicios del núcleo tales como el servicio de nombres y el de direcciones, así como también el servicio de administración de agentes, entre otros. El servicio de contexto el cual en un aspecto mínimo contribuye para controlar la evolución de las escenas por medio del contexto. Otras aportaciones para este caso de estudio fue la incorporación de módulos en la estructura de comunicación y control, entre los que podemos decir el de la generación de personalidades virtuales para agentes emocionales y el lenguaje denominado VEDEL ("Environment Description Language") [4.6], el cual nos auxilia para describir las escenas. Por otra parte se emplean AVE y LIA-3D [4.1] para mostrar los resultados. El caso de estudio presente nos permitió visualizar el reempeño de la aplicación para la mayoría de los módulos integrantes mencionados anteriormente.

En este caso de estudio se presentan dos tipos diferentes de entidades, los agentes que cumplen con el rol de depredadores y los agentes que tiene que evitar los ataques de

los depredadores, a esta clase de agentes los denominamos presas. La forma en la que el usuario puede definir las condiciones y el escenario inicial es por medio de un lenguaje semejante al natural, un ejemplo de cómo es que se puede definir un escenario por medio del uso de VEDEL:

```
[ENV]
        room.
[/ENV]
[ACTOR]
        whiteboy Antonio, north.
        brownboy Braulio, south.
        woman Carolina, front Braulio.
[/ACTOR]
[OBJECT]
        box, left Antonio.
[/OBJECT]
```

La Figura 4.2 nos muestra dos de los diversos actores con los que se cuenta en la plataforma dichos avatares son guiados por medio de comportamientos generados en agentes, a los cuales se les asigna un determinado comportamiento de acuerdo a lo que el usuario desea modelar. En la imagen presentada en la Figura 4.2 podemos observar dos avatares con comportamientos distintos los cuales son manipulados por agentes que presentan los comportamientos tanto de depredador como de presa respectivamente. Estos avatares son posicionados de acuerdo a las asignaciones iniciales proporcionadas por el usuario

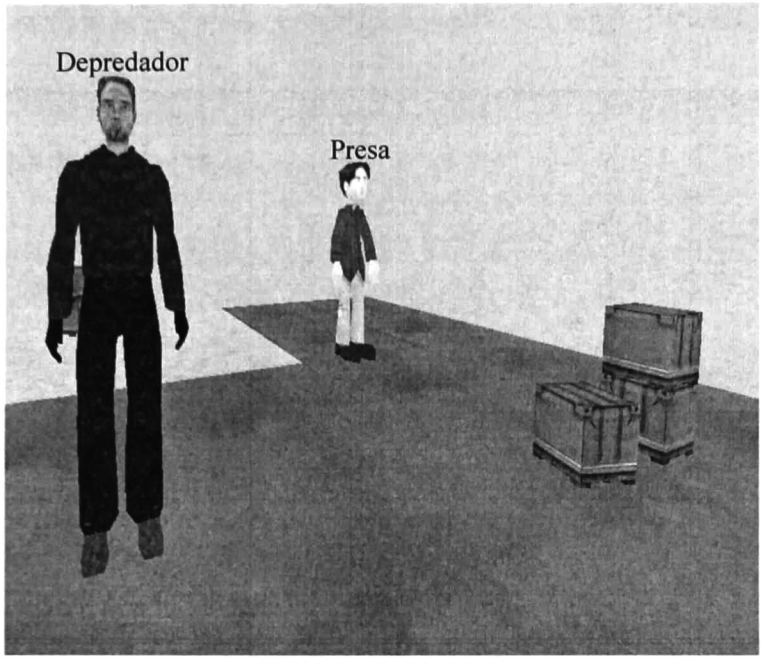


Figura 4.2 Condiciones iniciales

Una vez puestas las condiciones iniciales los agentes inician su ciclo de vida y el servicio de control de escena manipula la evolución de las diversas entidades regulando el accionar. Algunos de los servicios demandados para este caso de estudio son:

- El servicio de control de escena: Cumple con la coordinación de la interacción de las entidades virtuales conforme a las reglas generadas por el usuario.
- La capa de trasporte de mensaje sirve para la interconexión de los diversos módulos.
- El módulo de renderizado AVE-3D nos preemite proporcionar un visión más realista y en tres dimensiones de la evolución de la escena.

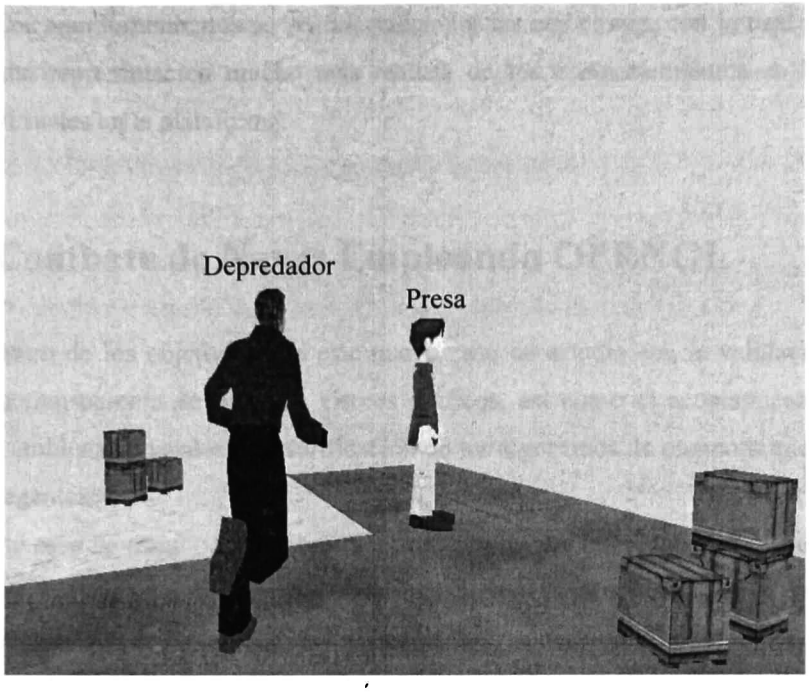


Figura 4.3 El agente a las izquierda en la imagen ataca al de la derecha

4.3.1 Resultados

Uno de los objetivos primarios para este caso de estudio (Depredador - Presa Empleado AVE-3D) es la utilización de la herramienta generada con anterioridad, al añadirla y adaptarla como parte de los posibles conexiones graficas, aprovechando con esto la capacidad que nos ofrece esta herramienta basada en AgentFX, algunas de los resultados importantes de este caso de estudio son los siguientes:

- Se logra orquestar la interacción entre las entidades, con total independencia de los algoritmos de comportamiento de las entidades u avatares.
- Se pueden modular y evaluar las acciones venidas por parte de cada una de las entidades en el ambiente.
- Se permite el control de ciclo de vida de las entidades.
- En las simulaciones se mantiene una pequeña base de datos con las características principales de todas las entidades participantes en la simulación.
- La representación grafica es de mejor calidad y en tres dimensiones, lo sentimientos en los avatares humanos son expresados de una manera gestual lo cual permite generar escenas con una mayor credibilidad para el usuario.

- Los comportamientos se ven afectados por las emociones, con lo cual se lograr una representación mucho más realista de los comportamientos en humanos virtuales en la plataforma.

4.4 Combate de Naves Empleando OPENGL

Dentro de los objetivos para este nuevo caso de estudio son la validación de la conexión transparente de diversos visores gráficos, así como el acoplamiento de los diversos ambientes visuales y la verificación de los algoritmos de comportamiento para algunos agentes.

Este caso de estudio trata sobre la interacción de dos diferentes tipos de entidades o avatares, los cuales están representado mediante naves de combate, cada una de las cuales representa dos diversos comportamientos, dichas naves participan en una simulación de combate que se desarrolla en una ambiente de varios cuartos en los cuales estas navegan o vagan intentando cumplir las metas asignadas para cada entidad o para los grupos de entidades.

La forma en como se define la simulación por parte del usuario es por medio de un lenguaje reducido semejante al natural denominado GOL [4.4, 4.7] el cual es empleado para describir las escenas y las metas de cada uno de los personajes en la simulación, es por este medio que se definen las condiciones iniciales para cada una de las entidades en el mundo virtual modelado. Un fragmento de un ejemplo de cómo se realiza una descripción empleado GOL es el siguiente:

Declarations

Ship2: XWing

Ship1, Ship3: Imperial

Behaviors

Ship2: Prey

Ship1, Ship3: Predator

Arrangement

Ship1: tends to the right-near corner of Kitchen

Ship2: in the Living Room, at most four meters from Ship1

Ship3: in the Kitchen, on the left side of Ship1
Sketch
Concurrent
Ship2 flies toward a place in the Dining Room // goal 1
Choice
Ship1 flies toward Ship2 // goal 2
Ship3 flies toward a place around two meters from Ship1 // goal 3
End-Choice
End-Concurrent
End-Sketch

La representación grafica de los dos tipos de naves se observan el la Figura 4.6 la cual denominamos nave "Imperial" y en la Figura 4.7 la nave enemiga que se nombro "XWing", así mismo en ambas imágenes se aprecia el mundo virtual modelado, donde se pueden ver los muros y puertas de los cuartos.

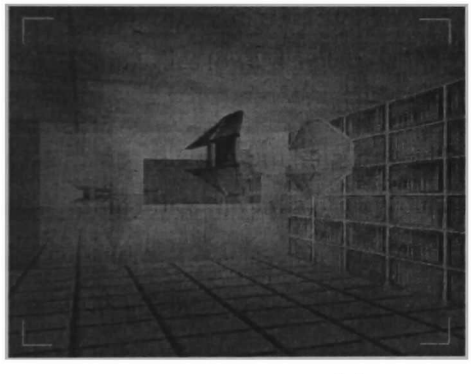


Figura 4.6 Nave Imperial

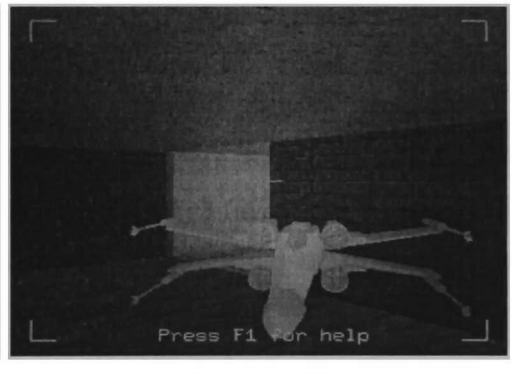


Figura 4.7 Nave XWing

La comunidad de agentes son controlados por el módulo de "Administración de Agentes" el cual se emplea para registrar las comunidades de agentes disponibles para las diversas aplicaciones.

Otro de los servicios empleados es el servicio de "Administración de Recursos" el cual se emplea para poder disponer de recursos de forma local o remota, permitiendo a cada uno de los componentes publicar y obtener recursos. Entre estos recursos se encuentras archivos de descripción de mundos virtuales, así como archivos de descripción de primitivas para la representación gráfica de entidades virtuales como

pueden ser otras naves u objetos virtuales que enriquezcan la capacidad de representación de la herramienta gráfica.

Finalmente este caso de estudio prueba la inclusión de habilidades a los agentes participantes en la simulación, por ejemplo un agente puede tener la habilidad de disparar y otro no, en la Figura 4.8 y Figura 4.9 vemos el accionar de disparo y el impacto del mismo en una nave. Con base en la participación de módulos como la "Comunidad de Agente" y servicios en el núcleo como son el servicio de "Administración de Recursos", "Administración de Agentes" y "GOL"

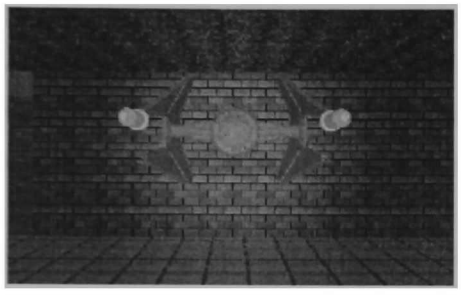


Figura 4.8 Acción de Disparo

Figura 4.9 Nave Impactada

4.4.1 Resultados

El objetivo principal de este caso de estudio fue el del empleo del servicio de "Administración de Recursos" y el servicio de "Representación del Conocimiento", los cuales son empleados por dos diferentes módulos del sistema GeDA-3D el primero de ellos es empleado por el módulo de visualización, la comunidad de agentes, entre otros, para el caso del servicio de "Representación de Conocimiento", este es utilizado por cada uno de los agentes para explotar y publicar el conocimiento generado por los mismo. Algunos de los resultados a destacar son los siguientes:

- La coordinación de los agentes y las acciones de los mismo, y la representación a medida de acuerdo a la descripción del usuario logrando el control de cómo se desea se realice la simulación.
- El utilización de recursos compartidos fue empleada con existo.
- Se permite que los agentes publique y generen conocimiento en una base de conocimiento mediante un motor de inferencia.
- Los algoritmos de comportamiento son el cuello de botella de la representación dado que no se emplean representaciones complejas para las entidades virtuales en los ambientes modelados, esto produce la necesidad de que el núcleo genere

búfer de manera temporal y la validación de las acciones pueden ser incongruentes.

4.5 Conclusiones

Es de destacar que en los últimos casos de estudio realizados sirvieron para evolucionar y mejorar las aproximaciones puramente teóricas y otras que no se emplean en ninguna otra investigación. Se realizan modificaciones al ciclo de ejecución-reacción [3.1], el cual se emplea para controlar la evolución de la escena con esta visión se logra determinar un realismo en desempeño de cada uno de los algoritmos de comportamiento. El uso de lenguajes semejantes al natural para la descripción de escenas nos permite facilitar el uso de este sistema a usuarios con poca experiencia.

Algunas de las aportaciones para el sistema son:

- La utilización de recursos compartidos entre los módulos.
- El empleo de bases de conocimiento pero en un sentido público o privado con pertenencia grupos.
- La utilización de servicio de recursos compartidos nos permite realizar aportaciones de nuevos escenarios, nuevos contexto y otro tipo de recurso empleados en la generación de escenarios virtuales en incluso se proyectaría para la generación dinámica de representación de entidades participantes en la simulación.
- El manejo y control de la evolución de escena por medio del uso de reglas dinámicas que rigen los efectos para cada una de las entidades en el ambiente.

Capítulo 5 Conclusiones

Resumen

En esta sección se presentan las conclusiones respecto a los objetivos planteados al inicio de este trabajo. También son presentados brevemente los puntos propuestos como líneas de investigación y trabajo futuro para el núcleo de GeDA-3D.

5.1 Introducción

El presente trabajo presenta el diseño e implantación apropiada de un núcleo o middleware para un sistema que maneja una interfase 3D. En este trabajo se presentaron los diferentes problemas que se necesitaron resolver para integrar los módulos que conforman el sistema GeDA-3D [1.1, 1.2, 1.3] que es el sistema que maneja una interfase 3D. Las aportaciones al núcleo de GeDA-3D son la implantación de servicios que forman parte del núcleo. El punto medular del trabajo es el intercambio de información entre los agentes para lograr coordinación y obtener mejores soluciones.

5.2 Resultados

Los aportes de esta investigación e estudio permiten al sistema realizar las siguientes tareas:

a) Administrar dinámicamente la manera en como los agentes interactúan para lograr sus metas y objetivos. En base a la generación de contexto y en base a este se norman los comportamientos y las consecuencias sobre las entidades de forma dinámica, siendo esto una aportación para la comunidad en cuanto al enfoque y empleo del ciclo modificado y adaptado de ejecución-reacción [3.1].
b) El servicio de administración de agentes norma la puesta en ejecución y paro de agentes de forma transparente por medio del servicio de administración de agentes. El servicio también realiza la correcta asignación de coordenadas, metas, habilidades y objetivos a las diversas entidades en el sistema.
c) Servicio de Administración de recursos: Este permite que los módulos compartan recursos como son archivos, estructuras, clases, definiciones de mundos, imágenes, etc.
d) Servicio de trasporte de mensaje: Permite abstraer las comunicaciones de la red subyacente y presenta una forma fácil y sencilla para envió y recepción de mensajes entre los módulos u agentes en la plataforma.
e) Servicio de Administración de Conocimiento: administra el conocimiento en dos conjuntos para mantener diferencias entre conocimiento público y conocimiento de pertenencia a grupos. Este servicio emplea la representación por medio de predicados sencillos que permiten que una maquina de inferencia genere nuevo conocimiento en base al conocimiento existente.

f) Control de Escena: Controla de una forma dinámica y practica la evolución de la escena, mediante la evaluación y conversiones en caso de ser necesario de cada uno de los comandos en formato ACL a otros formatos como pueden ser LIA u otros que sean necesario para que dicha acción sea representada en la pantalla del usuario. Si una valuación de las acciones se determina como incorrecta, entonces esta acción es ignorada y no es promovida hacia la representación gráfica.

Con lo anterior se cumplen los objetivos principales para este trabajo de investigación al posibilitar la comunicación y la integración de los diversos módulos que conforma el proyecto GeDA-3D. Esto se logra por medio del servicio de transporte de mensaje, útil en la comunicación y coordinación entre el módulo de representación gráfica y el servicio de control de escena entre otros. El servicio de control de escena regula la forma en como los agentes son afectados por el ambiente. El servicio de administración de recursos permite a los módulos compartir recursos como pueden ser tipos de ambientes, nuevas entidades graficas, representación diferentes de ambientes, archivos, etc.

5.3 Trabajo Futuro

En el desarrollo de cualquier trabajo siempre existen puntos a mejorar o implantar y estos deben ser considerados como trabajo futuro. A continuación hacemos una lista no exhaustiva de lo que consideramos como trabajo futuro:

- La administración de especificación de metas y habilidades, es aun una estructura sin explotar y presenta una serie de dificultades para ser administrada y para una correcta puesta en marcha. Este trabajo debe ser realizado.
- El servicio de trasporte de mensaje puede evolucionar en cuanto a la dinámica de envió y recepción de información, permitiendo envíos de información y manejos de protocolos más complejos para desarrollo de simulaciones con una mejor coordinación entre las entidades.
- Las consideraciones de seguridad actualmente no son considerados en la plataforma, y dado que la plataforma funcionará en ambientes hostiles, es necesario considerar seguridad a nivel del núcleo.

- La administración de conocimiento esta ahora limitada por los recursos de memoria, como una consecuencia de una estructuración no dinámica en el manejo de la representación de las reglas.

Apéndice A Conceptos Básicos

Resumen

Este apéndice presenta algunos de los temas teóricos fundamentales para la realización de la presente investigación.

A.1. Introducción

Este apéndice describe algunos de los temas relacionados con esta investigación, estos temas son descritos de una forma breve para brindar una introducción hacia el trabajo realizado en la presente tesis. Dichos temas como lo son los Sistemas Distribuidos, Sistemas Multi-agentes, guardan una relación directa y son paradigmas base para el desarrollo del proyecto denominado GeDA-3D. A continuación se presenta primero una breve descripción de los conceptos base y generales de los fundamentos de los Sistemas Distribuidos, se describen las características, también se describen los conceptos básicos de las redes de Petri y en particular se describe de forma concisa las redes de Petri coloreadas, así como los sistemas multi-agentes.

A.2 Sistemas Distribuidos

Los sistemas distribuidos (SD) pueden ser vistos como un grupo de componentes distribuidos geográficamente o en diferentes computadoras interconectados, capaces de compartir recursos y de procesar de forma cooperativa solamente por medio de paso de mensajes [A.1]. Las principales características de los Sistemas Distribuidos son descritas a continuación.

A.2.1 Características

Los sistemas distribuidos presentan como características principales las siguientes:

- *Compartir recursos*: El principal motivo para realizar un sistema distribuido es el de compartir los recursos, esto es poder acceder a ficheros o cualquier otro recurso que sea necesario compartir.
- *Apertura*: La extensibilidad de un sistema es la capacidad de incremento y re-implementación del mismo. Esto es si es posible añadir nuevos servicios y recursos para la utilización del sistema.
- *Concurrencia*: Un recurso puede ser accedido por múltiples usuarios, atendiendo una por una de las peticiones, de forma que se mantenga la persistencia de los datos.

- *Escalabilidad*: La escalabilidad es la capacidad del sistema es la capacidad de continuar efectividad ante un aumento considerable en el numero de recursos o usuarios.
- *Transparencia*: Es el ocultamiento de la separación de los componentes, con esto el usuario percibe que trabaja sobre un bloque sólido, en vez de un conjunto de componentes.
- *Tolerancia a Fallos*: Los fallos presentados dentro de los sistemas distribuidos pueden considerarse parciales, esto es algunos componentes del sistema fallan mientras otros continúan en operación. Lo que se pretende es ocultar el fallo
- *Seguridad*: La seguridad considera tres aspectos que son:
    - *Confidencialidad*: Sólo usuarios autorizados puede ver la información.
    - *Integridad*: Los datos no serán alterados por un tercero.

Algunas metodologías para desarrollar Sistemas Distribuidos por medio de comunicación asíncrona son empleados actualmente, entre ellos tenemos lo que es el empleo de la Invocación de Procesos Remotos o RPC por sus siglas en ingles, de este tipo de metodología una de las más comunes es *Java Remote Method Invocation* (RMI) [A.2] esta metodología a grandes rasgos permite que un objeto, invoque métodos de otro ejecutándose en una PC distante. Por tanto RMI nos ayuda al solucionar la comunicación entre programas, por otro lado la *Common Object Request Broker Architecture* (CORBA) [A.3] presenta una infraestructura abierta para objetos en cómputo distribuido, la cual intenta servir de estándar en el desarrollo de este tipo de sistemas.

A.3 Sistemas Multi-Agentes

Los Sistemas Multi-Agentes (SMA) es un tópico de la Inteligencia Artificial. Un SMA puede ser visto como un conjunto de agentes que tiene la habilidad y capacidad de trabajar de forma autónoma y cooperativa para desempeñar un objetivo en común y tareas que no podría realizar un solo agente.

A.3.1 Agente

No existe una definición para agente que sea empleada o aceptada de forma universal, diversos autores realizan sus propias definiciones. Por ejemplo para Wooldridge un agente es: "un sistema computacional que está situado en un ambiente y que es capaz de realizar acciones autónomas en esta ambiente con el propósito de lograr los objetivos para los que fue diseñado" la visión abstracta de lo agentes propuesta por Wooldridge se puede verse esquematizada en la siguiente Figura:

Figura A.1 Representación abstracta de un agente en su ambiente

Desde este punto de vista un agente es una entidad autónoma con la capacidad de percibir su entorno y por medio de acciones modificarlo, para conseguir algún objetivo asignado. Se considera también la coordinación entre los agentes para lograr un objetivo conjunto. Esta visión abstracta y general será adoptada para fines descriptivos y para fines prácticos en el concepto de agentes de esta investigación.

A.4 Redes de Petri

En esta sección se presentan los conceptos básicos de las Redes de Petri (RP) [2.6], y se presenta también un tipo especial de éstas las RP Coloreadas (RPC) que son una extensión importante de las RP generalizadas. Una estructura de RP G es un grafo dirigido bipartito, que puede ser visto como una 4-tupla G = (P, T, I, O) donde P = $\{p_1, \ldots, p_n\}$ es un conjunto finito de vértices llamados lugares, T = $\{t_1, \ldots, t_n\}$ es un conjunto finito de vértices nombrados transiciones, I: P x T $\rightarrow \mathbf{N}^+$ es una función que representa el peso de los arcos que van de los lugares a las transiciones, y O: P x T $\rightarrow \mathbf{N}^+$ es una función que representa el peso de los arcos que van de las transiciones a los lugares, $\mathbf{N}^+$ = {0,1,2, …, n}. Gráficamente los lugares son representados mediante círculos y las transiciones son representadas mediante rectángulos y por ultimo los arcos se

representan por medio de flechas, la Figura A.2 muestra una representación grafica de una RP.

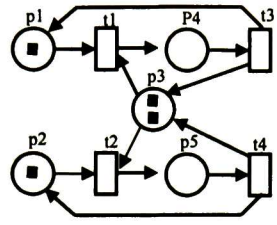


Figura A.2 Representación Gráfica de RP

A.4.1 Marcado

El marcado en una RP representa el estado del sistema, y sirve para ver y evaluar el comportamiento así como la evolución del sistema, gráficamente las marcas se representan por medio de puntos negros dentro de los lugares de la RP, cuando el número de marcas dentro de un lugar es grande este se expresa mediante un entero. A la distribución de marcas en un instante dado se denomina marcado [A.6].

Función de marcado M: $P \rightarrow \mathbf{N}^+$ la función de marcado M asigna a cada uno de los lugares de la red un número positivo incluido el cero, representado el número de marcas contenidas para cada uno de los lugares.

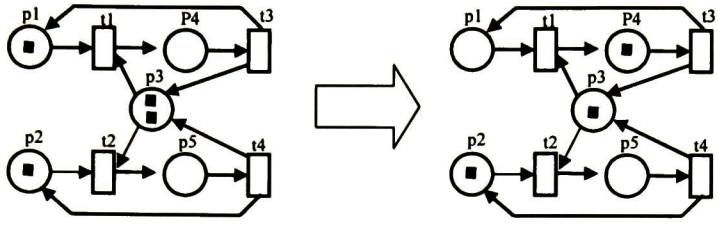
Transiciones habilitadas: Una transición "t" en un red de Petri esta habilitada en un marcado "M" si en los lugares de entrada a dicha transición existe por lo menos W (p_i, t) marcas: $\forall p \in \bullet t \quad M(p) \geq W(p, t)$, donde W representa la función de peso de los arcos. Una transición habilitada puede o no disparase si se encuentra habilitada, al igual un conjunto de transiciones habilitadas serán disparadas de forma no determinista de pendiendo de la metodología empleada, otro aspecto a destacar es que conforme se disparan las transiciones el número de marcas no siempre se conserva. Esto es un disparo de una transición "t" provoca la eliminación de W (p, t) marcas de de cada lugar de entrada "p" de la transición "t" y coloca una cantidad de W (t, p) marcas en cada uno de los lugares de salida de "p" de la transición "t" dando como resultado un cambio de marcado de un marcado M a uno M', lo cual puede ser expresado de la siguiente manera:

$$M'(p) = M(p) + W(t, p) \quad \forall p \in t\bullet \text{ y } p \notin \bullet t$$

$$M'(p) = M(p) + W(p, t) \quad \forall p \in \bullet t \text{ y } p \notin t\bullet$$

$$M'(p) = M(p)$$

Si una transición t esta habilitada en M y M' es la marcación resultante esta se representa: M $^{t}\rightarrow$ M'. En la Figura A.3 podemos ver la evolución de un marcado tras el disparo de la transición habilitada "t1".



A.3 Representación gráfica de la evolución de un marcado

A.4.2 Redes de Petri Coloreadas

Las RP Coloreadas (RPC) son una extensión de las RP generalizadas, fueron propuestas por K. Jensen en 1981 [A.4, A.5, A.6], algunas de las ventajas de estas sobre las generalizadas es que brindan un mejor nivel de abstracción, especialmente cuando dicho sistema puede ser expresado como un conjunto de componentes de comportamiento semejante, otro de los aspectos destacables es que los modelos se representan de forma muy condensada.

A.4.3 Conceptos Básicos de RPC

En las RPC las marcas que contiene cada uno de los lugares pueden ser diferenciadas entre si, cada marca puede pertenecer algún tipo (color) de dato que es representado por un identificador simbólico [A.4, A.5]. Los lugares tienen entonces la habilidad de contener diversos tipos (colores) y las transiciones la capacidad de trasmitirlos. Dichas capacidad o habilidad deberá ser formulada de forma explícita en el modelado.

A.4.4 Estructura de RPC

Una RPC es un grafo bipartito, que se puede representar por la quíntupla G = (P, T, C, I, O) donde P = $\{p_1, \ldots, p_n\}$ es un conjunto finito de vértices llamados lugares, T = $\{t_1, \ldots, t_n\}$ es un conjunto finito de vértices nombrados transiciones, C = {c1, c2, …, c_n} es un conjunto finito de colores o tipos de datos, es importante recalcar que también se pueden emplear todas las combinaciones posibles de los colores como nuevos colores.

Esto es C es el conjunto finito de colores asociados a los lugares y las transiciones, I: $C(T) \rightarrow$ sum $C(P)$ es una función que incidencia previa, y O: $C(P) \rightarrow$ sum $C(T)$ es una función de incidencia posterior. Estas funciones de incidencia están compuestas por las funciones asociadas a los arcos en la red $\forall p_i \in P$ y $\forall t_j \in T$:

$I(p_i, t_j)$: $C(t_j) \rightarrow$ sum $C(p_i)$ función asociada al arco (p_i, t_j)

$O(p_i, t_j)$: $C(t_j) \rightarrow$ sum $C(p_i)$ función asociada al arco $(t_j, p_i,)$

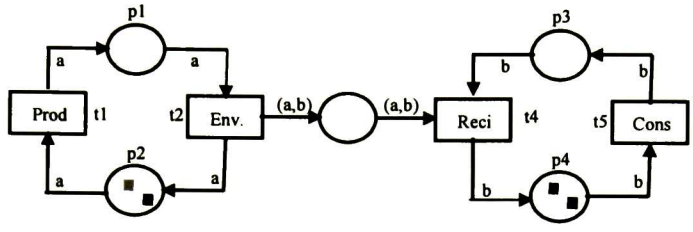


Figura A.3 Representación Gráfica de una RPC

A.5 Resumen

Se ha realizando un presentación de varios temas de interés para la presente investigación, se presentaron las características destacadas de los SD, una descripción somera de los SMA y por ultimo se toco el tema de las RP describiéndose primero las RP generalizadas y posteriormente las RPC, se hará uso del formalismo de RP especial mente la subclase de RPC para el modelado de algunos de los servicios desarrollados para el núcleo de GeDA-3D, todo esto debido a las ventajas y las facilidad de análisis que este tipo de subclase de RP, entre lo que podemos mencionar su facilidad de abstracción, la condensación de los modelos, y la naturaleza de representar mediante los colores (tipos).

Referencias

[1.1] MC Silvia Toscano Garibay. "Ambiente Genérico Virtual distribuido" 19 Septiembre – 2000

[1.2] H. Iván Piza, Fabiel Zúñiga, Félix R. Ramos. "A Platform to Design and Run Dynamic Virtual Environments". CyberWorlds-2004. November 2004, Tokyo, pp. 78-85 Japan ISBN 0-76952140-1

[1.3] Félix Ramos, David Garduño, Iván Romero, José Luis Márquez, "Arquitectura de una Oficina Virtual Distribuida", eMemories of the CIE98 congress, CINVESTAV México, 1998

[1.4] Félix Ramos, David Garduño, Iván Romero, José Luis Márquez, "A Multiagent Based architecture for Distributed Virtual Office", Memories of the Parallel and Distributed Processing Techniques and Applications PDPTA99, Las Vegas, USA, Jun 1999

[A.1] George Coulouris, Jean Dollimore, Tim Kindberg: Distributed Systems Concepts and Design, Thrid Edition
Addison Wesley Publishers Limited 1990, 1994, Pearson Education Limited 2001 ISBN 0-201-61918-0

[A.2] Java Remote Method Invocation (RMI) URL: http://java.sun.com/products/jdk/rmi/index.html

[A.3] Object Management Group, "Common Object Request Broker Architecture (CORBA): Core Specification", Version 3.0.3, formal/04-03-01 URL: http://www.omg.org o http://www.corba.org March 2004.

[A.4] K. Jensen. "Coloured Petri Nets – Basic Concepts, Analysis and Practical Use", Springer-Verlag, 2nd edition, 1992.

[A.5] K. Jensen. "Coloured Petri Nets – Basic Concepts, Analysis and Practical Use", Volume 2. Springer-Verlag, 1994

[A.6] Ernesto López Mellado. "Introducción a las Redes de Petri", Universidad Autónoma de Nuevo León, Octubre 1997.

[2.1] AgentBuilder User's guide, Reticular Systems, external documentation: URL: http://www.agentbuilder.com/

[2.2] Shoham Y., AGENT-0: "A simple agent language and its interpreter", In Proceedings of the Ninth National Conference on Artificial Intelligence, Vol II (pp. 704-709), Anaheim, CA, MIT Press, 1991.

[2.3] Shoham Y., "Agent Oriented Programming", Artificial Intelligence, 60(1), pp. 51-92, North-Holland, 1993.

[2.4] M. J. Huber. JAM: "A BDI-Theoretic mobile agent architecture". In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, Proceedings of the Third Annual Conference on Autonomous Agents (AGENTS-99), pages 236–243, New York, May 1–5 1999. ACM Press.

[2.5] S. R. Thomas, "PLACA, an Agent Oriented Programming Language", Ph.D. Thesis, Stanford University, 1993.

[2.6] Aglets: URL: http://www.trl.ibm.com/aglets/ o http://aglets.sourceforge.net/

[2.7] Alpha Works: URL: http://www.alphaworks.ibm.com/

[2.8] LANGE, Danny B.; OSHIMA, Mitsuru. "Programming and Deploying Java Mobile Agents With Aglets". Addison-Wesley Pub Co. 1998.

[2.9] A-Globe es un plataforma de agentes: URL: http://agents.felk.cvut.cz/aglobe/

[2.10] AMETAS: URL: http://www.accsis.com/ametas/index.html

[2.11] DIET Agents: URL: http://diet-agents.sourceforge.net/Index.html

[2.12] Howden, W., Ronnquist, R., Hodgson, A., Lucas, A.: JACK Intelligent Agents, URL: http://www.agent-software.com/shared/home/

[2.13] Busetta P., Rönnquist, R., Hodgson A., Lucas A., "JACK Intelligent Agents – Componenets for Intelligent Agents in Java", updated from AgentLink Newsletter, URL: http://www.agent-software.com.au/, October 1999.

[2.14] Mark d'Inverno, David Kinny, Michael Luck, and Michael Wooldridge, "A formal Specification of dMARS", 4th International Workshop on Agent Theories, Architectures, and Languages (ATAL'97), LNAI, Vol. 1365, pp. 155-176, Springer, 1998.

[2.15] F. Bellifemine, A. Poggi, G. Rimassa. "Developing multi agent systems with a FIPA-compliant agent framework". in Software - Practice & Experience, John Wiley & Sons, Ltd. vol no. 31, 2001, pagg. 103-128

[2.16] JADE Web Site: URL: http://jade.tilab.com/, http://jade.cselt.it/

[2.17] Telecom Italia: URL: http://www.telecomitalialab.com/

[2.18] FIPA: Foundation for Intelligent Physical Agents. http://www.fipa.org/

[2.19] Whitestein: URL: http://www.whitestein.com/pages/index.html

[2.20] Rockwell: URL: http://www.rockwell.com/

[2.21] Agent platforms which conform to the FIPA Specifications: http://www.fipa.org/resources/livesystems.html#jade

[2.22] Distributed Systems Group University of Hamburg, Germany http://vsis-www.informatik.uni-hamburg.de/projects/jadex/

[2.23] M. J. Huber. JAM: "A BDI-Theoretic mobile agent architecture. Third Annual Conference on Autonomous Agents" (AGENTS-99), pages 236–243, New York, May 1–5 1999. ACM Press.

[2.24] A. Rao and M. Georgeff. BDI-agents: from theory to practice. In Proceedings of the First Intl. Conference on Multiagent Systems, San Francisco, 1995.

[2.25] Jadex-Blackjack: URL: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/blackjack.php

[2.26] MadKit: URL: http://www.madkit.org/

[2.27] Gutknecht, O., Ferber, J., Michel, F.: "Integrating tools and infrastructures for generic multi-agent systems", 5th International Conference on Autonomous agents, Montreal, Quebec, Canada, ACM Press (2001)

[2.28] O. Gutknecht & J. Ferber, "Vers une méthodologie organisationnelle pour les systèmes multi-agents", JFIADSMA'99.

[2.29] Sycara, K., Paolucci, M., van Velsen, M., Giampapa, J.: "The Retsina MAS Infrastructure", Kluwer Academic Publishers (2001)

[2.30] Reusable Environment for Task-Structured Intelligent Network Agents (RESTINA): URL: http://www.cs.cmu.edu/~softagents/retsina_agent_arch.html

[2.31] Finin, T., Labrou, Y., Mayfield, J.: KQLM as an Agent Communication Language. Software Agents, MIT Press (1997)

[2.32] Sycara, K., Klusch, M., Widoff, S., Lu, J.: Dynamic Service Matchmaking Among agents in Open Environments. ACM SIGMOD Record 28(1) (1999)

[2.33] Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press (1998)

[2.34] Cheyer, A., Martin, D.: "The Open Agent Architecture. Journal of Autonomous Agents and Multi-Agent Systems", 4(1) (2001)

[2.35] Nwana, S., Ndumu, D.T., Lee, L.C., Collis, J.C.: Zeus: A Toolkit for Building Distributed Multi-Agent Systems. 3th International Conference on Autonomous Agents, Seattle, WA, USA (1999)

[2.36] Zeus: URL: http://193.113.209.147/projects/agents/zeus/

[2.37] CHAUHAM, D.; BAKER, A. JAFMAS: "A Multiagent Application Development System". Proceedings of the Second International Conference on Autonomous Agents. 1998.

[2.38] Danny Weyns, H. Van Dyke Parunak, Fabien Michel, Tom Holvoet, and Jacques Ferber: "Environments for Multiagent Systems State-of-the-Art and Research Challenges": E4MAS 2004, LNAI 3374, pp. 1–47, 2005. © Springer-Verlag Berlin Heidelberg 2005.

[2.39] Scalable Fault Tolerant Agent Grooming Environment URL: http://sage.niit.edu.pk/About_SAGE.htm

[2.40] INEVAI3D: Integración de Escenarios Virtuales con Agentes Inteligentes 3D (TIN2004-07926)., I CONGRESO ESPAÑOL DE INFORMÁTICA (CEDI'2005), Simposio de Computación Ubicua e Inteligencia Ambiental.

[2.41] El proyecto INEVAI 3D: Agentes Autónomos 3D, Escenarios virtuales e Interfaces Inteligentes para aplicaciones de Domótica y de Realidad Virtual URL: http://www.inevai3d.com/

[2.42] Nidd, Michael. "Timeliness of Service Discovery in DEAPspace. IBM Research", Zurich Laboratory. Proceedings of the 2000 International Workshops on Parallel Processing (ICPP'00 Workshops). 0-7695-0771-9/00. 2000 IEEE.

[2.43] Virtual Reality Lab URL: http://ligwww.epfl.ch/Projects/projects_index.html

[2.44] AréVi : Atelier de Réalité Virtuelle URL: http://www.cerv.fr/fr/activites/AreVi.php

[2.45] ARéViJava : une plateforme de réalité virtuelle pour les visites guidées de sites patrimoniaux. S. Morvan, E. Maisel et J. Tisseau. Séminaire Maquette virtuelle et patrimoine - Cluny. pp. 45-48 March 2003

[2.46] oRis-ARéVi : un environnement de développement pour l'autonomisation des modèles. F. Harrouet. Tèmes Journées du Groupe de Travail Animation et Simulation - Bordeaux. June 2002

[2.47] P. A. Bernstein, "Middleware: A Model for Distributed System services", Communications of the ACM, vol. 39, pp. 86-98, February 1996.

[2.48] Subbu Allamaraju, "Nuts and Bolts of Transaction Processing", URL: http://www.subbu.org/articles/transactions, September 2005

[2.50] "Application Architecture for .NET: Designing Applications and Services" URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/distapp.asp, December 2002.

[2.51] T. Kramp and R. Koster "A Service-Centred Approach to QoS-Supporting Middleware", IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing Lake District, UK September 1998.

[2.52] Wolfgang Emmerich, "Software Engineering and Middleware: A Roadmap", International Conference on Software Engineering,

Limerick, Ireland, Pages: 117 - 129 ISBN:1-58113-253-0 2000.

[3.1] Weyns, D., Holvoet, T.: "A Formal Model for Situated Multi-agent Systems" Formal Approaches for Multi-Agent Systems, Special Issue of Fundamental Informatics, pp. 63-65, 2004

[3.2] H. Iván Piza, Fabiel Zúñiga, Félix R. Ramos. "GeDA-3D a Middleware useful to handle the evolution in behavioral animation-based virtual worlds with a multi-agent architecture", Innovative Internet Community Systems 4th International Workshop, IICS 2004 . Guadalajara, México , June 21-23, 2004

[4.1] MC Alma Verónica Martínez González "Lenguaje para Animación de Criaturas Virtuales". Agosto – 2005

[4.2] OpenGL. Desarrollado por Silicon Graphics. URL: http://www.sgi.com/software/opengl/

[4.3] H. Iván Piza, Fabiel Zúñiga, Félix R. Ramos. "A Platform to Design and Run Dynamic Virtual Environments". CyberWorlds-2004. Noviembre 2004, Tokyo, pp. 78-85 Japan ISBN 0-76952140-1

[4.4] H. Iván Piza, Fabiel Zúñiga, Félix R. Ramos. "Using a Declarative Language to Describe the Interactions in Virtual Scenes". 11th International Conference on Parallel and Distributed Systems (ICPADS'05). Fukuoka, Japan. Julio 2005, pp. 224-229. ISBN ~ ISSN: 521-9097, 0-7695-2281-5.

[4.5] AgentFX es una herramienta avanzada para el desarrollo de aplicaciones 3D interactivas URL: http://www.agency9.se/

[4.6] Alonso Aguirre, Luis Razo, Jaime Zaragoza "Caso de Estudio Básico (Combate de ranas)" URL: http://intranet.gdl.cinvestav.mx/sd/images/stories/files/applet.htm

[4.7] Fabiel Zuñiga, Félix Fco. Ramos, Hugo Iván Piza. "Specifying Agents Goals in 3D Scenarios Using Process Algebras", International Symposium of Advanced Distributed Systems (ISSADS-2005). Lecture Notes in Computer Science, p. 472. publicado: Springer Berlin / Heidelberg. Guadalajara, Mex. Enero 2005. ISBN: 3-540-28063-4.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Núcleo de GeDA-3D

del (la) C.

Alonso AGUIRRE GUTIERREZ

el día 02 de Febrero de 2007.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 2B
CINVESTAV Unidad Guadalajara

Dr. Mario Angel Siller González
Pico
Investigador CINVESTAV 2A
CINVESTAV Unidad Guadalajara