BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE COMPUTACIÓN

DISEÑO E IMPLEMENTACIÓN DE UN AGENTE INTELIGENTE DENTRO DEL MUNDO VIRTUAL SECOND LIFE

TESIS PROFESIONAL
PARA OBTENER EL TITULO DE

INGENIERO EN CIENCIAS
DE LA COMPUTACIÓN

PRESENTA:

ISRAEL GUZMÁN PÉREZ

ASESOR DE TESIS:
DRA. DARNES VILARIÑO AYALA

COASESOR DE TESIS:
DRA. FABIOLA LÓPEZ Y LÓPEZ

PUEBLA, PUE. ABRIL 2010

Agradecimientos:

A mi mamá Raquel Esperanza Pérez García…

Por tus sacrificios, por tu apoyo, por confiar en mí por el gran ejemplo de trabajo y esfuerzo constante que siempre me has demostrado.

A mi asesora Dra. Darnes Vilariño Ayala…

Por su asesoramiento y estimulo para aprender más y para terminar satisfactoriamente este proyecto.

RESUMEN

En el presente trabajo se propone el diseño e implementación de un agente inteligente dentro del mundo virtual Second Life, con la finalidad de que interactúe con avatars manejados por humanos. Al agente se le dota de la capacidad de enseñar las operaciones básicas (suma, resta, multiplicación y división) sobre un ábaco.

Para que logre interactuar con los usuarios dentro del mundo virtual, se le ha desarrollado un módulo de de procesamiento de lenguaje natural (PLN), un módulo de movilidad, un módulo de razonamiento y uno de procesamiento visual.

El lenguaje utilizado para el desarrollo del agente ha sido lenguaje Visual C++ y Linden Script Language, y se ha programado sobre el visor de Second Life.

TABLA DE CONTENIDO

INTRODUCCIÓN

Los mundos virtuales cada día adquieren más popularidad, para muchos individuos el mundo virtual se ha convertido en un lugar de trabajo, de adquisición de conocimiento y de esparcimiento. Las Universidades se han incorporado a ello de manera inmediata ofreciendo a sus profesores y estudiantes espacios comunes para el desarrollo del proceso de Enseñanza-Aprendizaje. En este trabajo se propone el diseño e implementación de un agente capaz de manejar a un avatar sobre la plataforma Second Life [4]. El agente denominado Israel debe ser capaz de dotar de habilidades a un avatar (Israel) para que enseñe el manejo de las operaciones básicas del ábaco a otros avatars, los otros avatars pueden estar manejados por humanos o por agentes. Este avatar debe poseer la capacidad de ver, moverse, escuchar y aprender.

Second Life (SL) es un metaverso lanzado en el año 2003, desarrollado por Linden Research Inc., el cual ha tenido una atención internacional de manera creciente desde el año 2006, se ha inspirado en la novela de Ciencia ficción “Snow Crash”, de Neal Stephenson y el movimiento literario “Cyberpunk”[1]. Las personas para hacer uso de éste programa, deben crear una cuenta en [www.secondlife.com] y bajar el programa llamado Second Life Viewer. Al registrarse y acceder pasarán a ser llamados “residentes" o de manera abreviada AV que significa avatars.

La manera en que los residentes interactúan a través de SL, lo cual a su vez es uno de los principales atractivos de este mundo virtual, es a través de los avatars o AV, que son personajes en 3D completamente configurables, lo que le da a los usuarios la capacidad de convertirse en otra persona y gozar (como el mismo nombre del programa indica) de una segunda vida. Los residentes de SL podrán explorar el mundo, conocer a otras personas, socializarse, participar en actividades grupales de acuerdo a sus gustos, tener sexo virtual, entre otras cosas. Su segundo atractivo más importante es la posibilidad de crear objetos e

intercambiar diversidad de productos virtuales a través de un mercado abierto que tiene como moneda local, el Linden Dólar ($L) [4]. Algo importante a destacar es que esta moneda virtual puede ser convertida a dólares de verdad, la cotización está aproximadamente (esto puede variar) 1 Dólar Estadounidense es igual a 250 Linden Dólar.

En estos momentos muchas Universidades ya han incorporado sus servicios dentro del mundo virtual y desarrollan gran parte de sus cursos y asesorías dentro del mismo. Por otra parte Empresas importantes como IBM, Microsoft, Colgate, Palmolive, Samsung, Disney, Kraft, Toshiba, Ford entre otras realizan ventas y acciones de publicidad dentro del mundo virtual [11].

La Teoría de Agentes ha alcanzado gran auge en los últimos años y se está incorporando de manera directa en el proceso de Enseñanza – Aprendizaje, en las simulaciones financieras y de mercado TAC CAT y TAC SCM [2], en la recuperación de información entre otras. En los últimos años se ha intentado incorporar el desarrollo de software inteligente a los mundos virtuales y surge con ello uno de los proyectos más ambiciosos de los últimos años, el proyecto Edd Hifeng [3], donde un grupo de investigadores de Rensselaer Plutechnic Institute están trabajando en la creación de un agente capaz de tener creencias y razonar acerca de las creencias de otros. Las características del agente son que puede predecir y manipular el comportamiento de otros avatars (jugadores humanos) con los cuales interactúa directamente. En una reciente conferencia sobre inteligencia artificial [5], los investigadores dieron a conocer sus logros en este proyecto: “Eddie”, un niño de cuatro años en Second Life quien puede razonar acerca de sus propias creencias y dar sus propias conclusiones, de la misma manera en que lo haría un niño humano de cuatro años.

Continuando con esta línea de investigación en el presente trabajo se tiene por objetivo el diseño e implementación de un agente inteligente, que maneje un avatar, capaz de enseñar las operaciones básicas sobre un ábaco (suma, resta, multiplicación y división), a otros avatars dentro de Second Life. Este agente posee 4 módulos básicos, para generar en el avatar la capacidad de

comunicación, visibilidad, movilidad y la interacción con los otros avatars dentro del mundo virtual.

En el capítulo uno se definen todos los aspectos teóricos necesarios para comprender como se van a llevar a cabo los objetivos de la tesis, en el capítulo dos se describe formalmente el problema y se propone el diseño del agente, en el capítulo tres, se muestran los algoritmos empleados para la implementación del agente, además de los requerimientos necesarios. En este mismo capítulo se señalan los resultados obtenidos; posteriormente se brindan las conclusiones y el trabajo a futuro. Al final se exponen dos apéndices, uno correspondiente a la instalación y configuración del visor de Second Life y el otro apéndice correspondiente a la instalación de Opensim.

Capítulo 1 Aspectos Teóricos

En el presente capítulo se discuten todos los aspectos teóricos necesarios que permiten sustentar el trabajo de investigación desarrollado, en particular se define el concepto de agentes y los términos inherentes al mismo, la forma en que se maneja el ábaco, se discute sobre los mundos virtuales y las herramientas necesarias para el desarrollo del agente.

1.1 Agentes

La tecnología de agentes se está usando para abordar la solución de múltiples problemas que se están presentando en la actualidad dentro de las Ciencias Computacionales, ya que un agente se puede definir como un sistema informático, situado en un entorno, en el cual sea capaz de realizar acciones flexibles y autónomas para alcanzar sus objetivos.

Los agentes en general poseen características tales como [2]:

***Autonomía*:** Un agente es capaz de realizar una acción autónoma flexible en algún ambiente.

***Reactivo*:** Reacciona a los cambios en el medio ambiente. Mantiene una interacción en curso con su medio ambiente, y responde a los cambios que ocurren en él.

***Pro-Activo*:** Genera y trata de llegar a sus metas, no es conducido solamente por los acontecimientos, toma la iniciativa y también reconoce oportunidades.

Habilidades Sociales: Interacción con otros agentes. Capacidad de comunicarse con otros agentes, por medio de algún lenguaje de comunicación.

Las características de las ***Habilidades Sociales*** se pueden clasificar como sigue:

- Cooperación.
- Competencia.

- Negociación.

1.2 Mundos Virtuales o Metaverso

El término metaverso fue acuñado por el escritor Neal Stephenson en su novela de ciencia ficción Snow Crash (1992). Stephenson, construyó, a partir de una hipótesis (un universo dentro de un universo), una compleja sociedad virtual que, increíblemente, se parece mucho a Second Life. ¡Y casi 15 años antes! En Snow Crash, el metaverso es un mundo virtual al que cualquiera puede entrar y que puede ser accedido desde cualquier terminal, pública o privada, en cualquier lugar del mundo. La fascinación con este mundo virtual no tiene precedentes y casi no hay persona que no tenga una segunda vida, pero dentro del metaverso hay mucho del mundo real. La discriminación se cuela por entre los bits y trasciende las barreras de la realidad. Sucede que, al ingresar desde una terminal pública, la calidad y definición del avatar es menor y, por lo tanto, reconocible como un ser de poco status. Lógicamente, aquellos que acceden por terminales públicas son los que menos recursos tienen en la vida real. Son personas que, impotentes, descubren que ni la virtualidad los escuda ante sus carencias.

En el metaverso el status se gana igual que en el mundo real. Los que tienen mayores recursos pueden comprar todo tipo de cosas y pueden entrar a zonas restringidas (como el famoso club Black Sun). Aquellos con gran conocimiento técnico también son respetados porque, generalmente, se muestran con avatares sofisticados hechos por ellos mismos.

1.2.1 Diversas Aplicaciones

Aparte de los simuladores de vuelo y otras aplicaciones de este tipo que sirven desde hace años para la enseñanza y la práctica de determinados oficios, existen ya programas que, mediante cámaras de vídeo y software permiten construir «puertas virtuales» que enlazan un despacho u oficina con otro punto cualquiera del edificio (por ejemplo, un pasillo) y permiten a los que pasan por éste, ver y

hablar con los que están en el despacho, estableciendo comunicaciones bidireccionales arbitrarias, no previstas por el arquitecto.

En la actualidad aparte del desarrollo que están llevando a cabo diferentes universidades en este campo el grueso de la investigación en la generación de entornos virtuales que traten de reproducir la realidad está siendo llevada por las empresas de ocio electrónico, que ven en esta tecnología una salida para sus videojuegos.

Mediante una tarjeta aceleradora 3D que se puede adquirir en cualquier tienda de componentes informáticos (por un precio entre 70 a 250€) una computadora adquiere la potencia necesaria para reconstruir en tiempo real verdaderos entornos tridimensionales que reflejen un tipo de paisaje, ya sean ciudades, selvas o el espacio. El grado de realismo conseguido sobrepasa lo imaginado hace una década y año tras año esta tecnología se duplica, lo que hace muy factible pensar que en menos de cinco años todos dispongamos de auténticos dispositivos para introducirnos en mundos virtuales en nuestro propio hogar.

Realmente todos los videojuegos se desarrollan sobre mundos virtuales, pues sus universos son simulaciones de la realidad o de diversos escenarios de ficción, pero virtuales.

Los mundos virtuales están de moda, y jamás habían generado tanta atención. Su popularidad no se limita a los tradicionales juegos de rol, que siempre han tenido una base importante de adeptos, sino que en los últimos meses se ha visto un gran crecimiento de las comunidades virtuales, que se controlan por avatares y que permiten relacionarse con otras personas.

Para esclarecer el estado de este sector, en GigaOM[16](esta lista de GigaOM es sólo temporal debido a que las cifras en los mundos virtuales cambia cada mes. Esta lista corresponde al año 2007, las cifras actuales correspondientes a Second Life se exponen en capítulos posteriores) , han elaborado un ranking con los 10

MMOs (Massively Multiplayer Online worlds) más populares, en términos de usuarios activos o suscritos:

1. World of Warcraft, lanzado en 2004 y con 8,5 millones de suscriptores. Estos números colocan a WoW en la primera posición de este ranking, en gran parte gracias a China, ya que el país asiático contribuye con 4 millones de jugadores.

2. Habbo Hotel, lanzado en 2000 y con 7,5 millones de usuarios activos. Este mundo virtual tiene su base en Finlandia y es muy popular entre los adolescentes.

3. RuneScape, lanzado en 2001 y con 5 millones de usuarios activos. Esta comunidad está implementada en lenguaje Java y da la posibilidad de elegir entre una cuenta gratis y otra de pago.

4. Club Penguin, lanzado en 2006 y con 4 millones de usuarios activos. Se trata de un MMO para niños cuyos juegos son similares a los de otras redes, como Habbo Hotel.

5. Webkinz, lanzado en 2005 y con 3,8 millones de usuarios activos. Webkinz también va dirigido a un público infantil y se caracteriza por su gran originalidad, que le ha permitido colocarse entre los cinco primeros.

6. Gaia Online, lanzado en 2003 y con 2 millones de usuarios activos. No es del todo un MMO ni tampoco un sitio social, pero parece ser que están pensando enfocarlo finalmente como un sitio para jugar en red.

7. Guild Wars, lanzado en 2005 y con 2 millones de usuarios activos. Se trata de otro MMO, aunque este no admite usuarios de Mac, cosa que sí hace RuneScape.

8. Puzzle Pirates, lanzado en 2003 y con 1,5 millones de usuarios activos. Lo publica Ubisoft y lo ha desarrollado indy king Three Rings; reúne a todos aquellos jugadores interesados en navegar los siete mares y ser el más temible de los piratas.

9. Lineage I/II, lanzado en 1998 y con 1 millón de suscriptores. Aunque en su día Lineage fue el más popular, los cambios que se han ido introduciendo y la aparición de otros mundos virtuales han hecho que haya ido cayendo.

10. Second Life, lanzado en 2003 y con 500.000 usuarios activos. Después de estar apareciendo constantemente en los medios desde su creación, Second Life ya no necesita ninguna presentación y seguramente su cifra de usuarios activos aumentará como la espuma en los próximos meses.

1.2.2 Second Life

Second Life (abreviado como SL) (en) / Segunda Vida (abreviado como SV) (es) un metaverso lanzado en el año 2003, desarrollado por Linden Research Inc. (llamado comúnmente Linden Lab), el cual ha tenido una atención internacional de manera creciente desde el año 2006[17]. Las personas para hacer uso de éste programa, deben crear una cuenta en [www.secondlife.com] y bajar el programa llamado Second Life Viewer. Al registrarse y acceder pasarán a ser llamados “residentes" o de manera abreviada AV que significa avatars.

La manera en que los residentes interactúan a través de SL, la cual a su vez es uno de los principales atractivos de este mundo virtual, es a través de los avatars o AV, que son personajes en 3D completamente configurables, lo que le da a los usuarios la capacidad de convertirse en otra persona y gozar (como el mismo nombre del programa indica) de una segunda vida. Esto promueve en el mismo mundo una avanzada interacción virtual que les permite a los residentes de SL explorar el mundo, conocer a otras personas, socializarse, participar en actividades grupales de acuerdo a sus gustos, tener sexo virtual, entre otras cosas. Su segundo atractivo más importante es la posibilidad de crear objetos e intercambiar diversidad de productos virtuales a través de un mercado abierto que tiene como moneda local el Linden Dólar ($L).

SL es uno de los varios mundos virtuales inspirados en la novela de Ciencia ficción “Snow Crash”, de Neal Stephenson y el movimiento literario “Cyberpunk”. Es un mundo creado por sus usuarios en el que la gente puede interactuar, jugar,

comunicarse y también hacer negocios, pues la manera en que se realizan negocios con la moneda Linden Dólar (Linden o $L) es abierta y libre a las interacciones del mercado. Esta moneda es intercambiable al mundo real, por lo que muchos residentes de SL se toman este mundo muy en serio convirtiéndolo en su sustento para la vida real.

En ocasiones, SL se ha definido como un juego online, lo que hace a esta definición corta por no tratarse de conquistar mundos, de obtener records, de pasar niveles o de crear estrategias, debido a que en SL no hay ganadores ni perdedores, sino que se puede interactuar con otros residentes o avatars en distintas actividades, entre ellas, juegos, como batallas con armas, partidos de fútbol, sexo, etc

Para marzo de 2008, SL cuenta con aproximadamente unos 13 millones de personas registradas, de las cuales un alto porcentaje están inactivas. La razón más común es que los interesados se registran, bajan el programa, pero el mismo no les permite arrancar, debido a que el software pide estándares en promedio altos para su ejecución. Además, hay que mencionar que muchas personas tienen múltiples cuentas, con el fin de desenvolverse con distintos roles en SL o hacer transacciones dudosas en el mundo virtual. Aun así, en promedio están conectados entre 35 mil a 50 mil personas y en sus picos más altos pueden llegar a estar de 70 mil a 90 mil conectados. Para la última estadística de ingresos en los últimos 60 días del año 2009, se conectaron 1,292,114 personas.

La programación de este mundo virtual es abierta y libre. El código de SL permite a los usuarios modificar absolutamente cualquier aspecto del mundo virtual, desde el color de los ojos del personaje hasta su aspecto físico, sus movimientos, sonidos y permite además, construir cualquier cosa en 3D: desde un cubo a una discoteca, un jardín o un campo de batalla o desde una pistola a una flor o unas zapatillas Nike. También permite la creación y manipulación de scripts para programar cualquier aspecto del mundo, desde un cañón para lanzar personas (como en el circo) hasta un sistema de envío de mensajes a móviles en cualquier lugar del mundo. Además de permitir editar todos estos aspectos, la propiedad

intelectual de los mismos pertenece al usuario que lo creó, por lo que legalmente puede obtener beneficios económicos, ya sea desde la moneda del mundo $L o tramitar sus ganancias a una cuenta corriente o de PayPal para obtener euros (€) o dólares ($).

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

Muchas universidades y empresas están utilizando Second Life para la formación, incluyendo las universidades de Harvard, Oxford, Universidad de Puerto Rico y Vigo[18].

En el 2007 se empezó a usar Second Life para la enseñanza de idiomas. La enseñanza de inglés como un idioma extranjero ha conseguido una presencia a través de varias escuelas, incluyendo el British Council, que ha tenido un enfoque en 'Teen Grid' (la versión de Second Life para adolescentes). Además, el Instituto Cervantes de España tiene también una isla en Second Life[18]. Se encuentra una lista más completa de proyectos educativos en Second Life (incluso algunas escuelas de idiomas)[18]. En la página Web de SimTeach. SLanguages 2008 es la segunda conferencia anual para la educación de idiomas utilizando los mundos virtuales como Second Life[19].

1.3 Herramientas de Programación

La aplicación utilizada para conectarse a SL es el Second Life Viewer, ésta aplicación es de código abierto por lo cual se puede modificar lo que se desee, pero sin embargo es necesario la utilización de varias herramientas [20] como son:

- LSL (Linden Script Language). Lenguaje default para programar dentro de SL.

- Visual C++ Express o Profesional 2008: Utilizado para programar el Second Life Viewer, el SL Viewer está implementado en esta herramienta, pero utiliza herramientas de terceros.

- Python 2.6: Permite realizar una precompilacion del Second Life Viewer.
- CMake 2.4: Utilizado junto con Python para realizar la precompilacion.
- Cygwin 3.2.39: Permite instalar las herramientas patchutils, flex y bison.
- Windows SDK for Windows Server 2008 and .NET Framework 3.5: Es utilizado para todo el desarrollo de la tecnologia .NET.
- DirectX 9.0 SDK: Se utiliza para el manejo de los gráficos.
- FMOD 3.75 API For Win32: Utilizado para el procesamiento de sonido.
- Quicktime SDK for Windows 7.1: Utilizado para el procesamiento de video.

  Además son necesarias las siguientes librerías: Boost Hell 1.31, apr_suite-1.2.8, ares-1.3.0, curl-7.16.0, expat-1.95.8, freetype-2.1.5, glh_linear, GL-windows, gtk-atk-pango-glib-windows, jpeglib-6b-windows, libpng-1.2.18-windows, llmozlib-windows, mesa-7.0, ndofdev-windows, ogg-vorbis-1.03-1.1.2-windows, openjpeg-1.2-windows, openSSL-0.9.7c-windows, SDL-1.2.5-windows, vivox-windows, xmlrpc-epi-0.51-windows y zlib-1.1.4-windows.

Cada una de estas herramientas son necesarias para la compilación exitosa del SL Viewer, pero para llevar a cabo la implementación del agente en el SL Viewer además son necesarias las siguientes herramientas:

- Servidor Apache 2.2.9: Utilizado para revisar páginas web.
- phpMyAdmin 2.11.2.9: Utilizado para revisar manualmente la BD de MySQL.
- MySQL 5.0.67: Almacena la BD de Conocimiento del Agente.
- PHP 5.2.6: Utilizado para consultas en MySQL.

- MySQL Connector C++ 1.0.4.0: Se utiliza para la conexión del Agente a la BD.
- Visual C++ Express o Profesional 2008: Es utilizado para programar al Agente.
- eSpeak 1.43: Se utiliza para dotar al agente de una voz.
- solicitudgetCLR 1.0: Permite realizar solicitudes http (desarrollado por Israel Guzmán Pérez).
- Avimator 1.0: Permite crear animaciones del avatar.
- QAvimator 1.0: Utilizado para crear animaciones del avatar.

1.3 Lenguaje Unificado de Modelado (UML)

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Se aplica en el desarrollo de un software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, sólo se modela mediante diagramas la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, sin embargo, la programación orientada a objetos viene siendo un

complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

1.4 Swanpan o Abaco Chino

La Aritmética del Ábaco (Ver Figura 1.1) es un determinado método de cálculo en el que los números están representados por bolas de madera. Estas bolas están sistemáticamente colocadas en una tablilla conocida con el nombre de Ábaco Chino. El término Aritmética del ábaco se usa para distinguirla de otros tipos de aritmética, en los que se usan representaciones escritas. Se le podría denominar ciencia de los números, pero ya que se usa comúnmente en la vida comercial, es más apropiado hablar de ella como arte del cálculo.

1.4.1 Construcción del ábaco.

El ábaco, tal como aparece en la figura 1.1, consiste en una tablilla rectangular de madera, dividida longitudinalmente en dos partes iguales por una varilla horizontal. Posee nueve, once, trece o más columnas de bolas móviles hechas generalmente de madera. El número de bolas en cada columna es de siete; dos por encima de la varilla horizontal y cinco por debajo de ella. A las bolas situadas en la parte superior de la varilla se las llama altobolas y a las que están situadas en la parte inferior se las llama hipobolas. Una altobola equivale a cinco hipobolas. Existe un tipo de ábaco que posee solamente seis bolas móviles en cada columna, una altobola y cinco hipobolas.

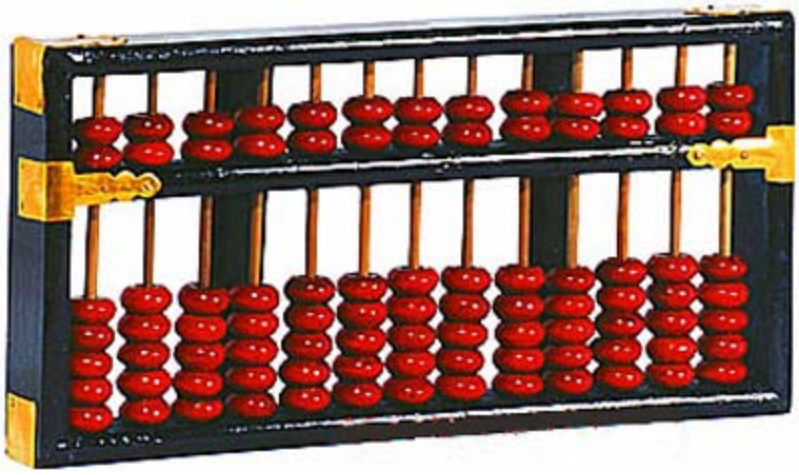


Figura 1.1 Ilustración del Swanpan o ábaco Chino.

El valor de la bola depende de la columna que se tome como unidad. Las bolas de la columna situada a mano izquierda tienen más valor que las situadas en la columna de la derecha. Una unidad de la columna del lado izquierdo posee diez veces más valor que su equivalente situada en la columna del lado derecho. De este modo si tomamos como unidad la primera columna del lado derecho de la varilla, una hipobola de la segunda columna tendrá el valor de diez unidades; una de la tercera columna será igual a cien unidades, etc.

1.4.2 Cómo usar el ábaco

Antes de empezar a usar el ábaco, todas las altobolas deben ser colocadas en el extremo superior de la tablilla y las hipobolas en el extremo inferior. Una vez colocadas así, están en disposición de ser movidas hacia arriba o hacia abajo para registrar cualquier número. La varilla de en medio es el eje al lado del cual se van colocando las bolas que vamos usando. Las bolas que permanecen inactivas o neutrales, deben ser colocadas en los lados.

Para sumar y restar no es necesario mover la altobola que está en el extremo superior ni la última de las hipobolas. Ya que una altobola equivale a cinco hipobolas, en vez de mover la última hipobola para contar hasta cinco podemos usar una altobola y devolver a la posición neutral(son los extremos del ábaco) o de inactividad a las cuatro hipobolas restantes. Del mismo modo ya que una hipobola

situada en la columna de la izquierda es igual a dos de las altobolas adyacentes situadas en la columna de la derecha, en vez de usar la altobola del extremo superior para sumar diez, podemos usar una hipobola de la columna izquierda y devolver a su posición neutral a la altobola del extremo inferior.

1.4.3 Suma

Se llama suma al proceso de combinar dos o más números dados para obtener uno igual a la totalidad de los mismos.

Guías para sumar:

Estas son las diecisiete guías o reglas para llevar a cabo la suma:

1. Uno — bajar cinco, anular cuatro
2. Dos — bajar cinco, anular tres
3. Tres — bajar cinco, anular dos
4. Cuatro — bajar cinco, anular uno
5. Uno — anular nueve, adelantar diez
6. Dos — anular ocho, adelantar diez
7. Tres — anular siete, adelantar diez
8. Cuatro — anular seis, adelantar diez
9. Cinco — anular cinco, adelantar diez
10. Seis — anular cuatro, adelantar diez
11. Siete — anular tres, adelantar diez
12. Ocho — anular dos, adelantar diez
13. Nueve — anular uno, adelantar diez
14. Seis — elevar uno, anular cinco, adelantar diez
15. Siete — elevar dos, anular cinco, adelantar diez
16. Ocho — elevar tres, anular cinco, adelantar diez
17. Nueve — elevar cuatro, anular cinco, adelantar diez.

Algunas de las palabras usadas necesitan ser explicadas. Por ejemplo bajar cinco significa desplazar hacia el eje central una altobola; anular significa sacar una bola que había sido antes colocada junto al eje central; elevar significa desplaza, hacia el centro una hipobola, adelantar diez significa desplazar hacia arriba una hipobola

de la columna de la izquierda. En cada guía, el primer número indica el número que ha de ser sumado a otro que está ya en una varilla, lo demás nos dice cómo proceder. Por ejemplo, el número cuatro está ya en la varilla de una columna, a este número le queremos sumar otro, el uno, ¿cómo proceder? Tomamos la primera guía: uno, bajar cinco anular cuatro, desplazamos hacia abajo en la misma columna cinco, es decir una altobola, y quitamos cuatro, es decir cuatro hipobolas.

1.4.4 Resta

Restar es el proceso por el cual averiguamos la diferencia entre dos números o bien averiguamos cuántos números debemos quitarle a un número determinado para obtener una parte del mismo. El número del cual ha de restarse otro se llama minuendo, y el número que ha de restarse de otro se llama sustraendo. Al resultado obtenido de tomar uno de otro se le llama residuo.

Guías para restar:

1. Uno — anular cinco, devolver cuatro.
2. Dos — anular cinco, devolver tres.
3. Tres — anular cinco, devolver dos
4. Cuatro — anular cinco, devolver uno.
5. Uno — anular diez (sacar una hipobola de la columna de la izquierda), devolver nueve.
6. Dos — anular diez, devolver ocho.
7. Tres — anular diez, devolver siete.
8. Cuatro — anular diez, devolver seis.
9. Cinco — anular diez, devolver cinco.
10. Seis — anular diez, devolver cuatro.
11. Siete — anular diez, devolver tres.
12. Ocho — anular diez, devolver dos.
13. Nueve — anular diez, devolver uno.
14. Seis — anular diez, devolver cinco, anular uno.
15. Siete — anular diez, devolver cinco, anular dos.
16. Ocho — anular diez, devolver cinco, anular tres
17. Nueve — anular diez, devolver cinco, anular cuatro.

En cada guía el primer número es el número que ha de ser restado, y lo que sigue nos indica cómo proceder. Por ejemplo en la primera guía: Uno anular cinco, devolver cuatro, el número uno es el que ha de restar de otro número que está ya en la varilla (minuendo). Sabemos que cinco menos uno es cuatro, pero ¿cómo vamos a restar en la tablilla? La guía nos habla de cancelar cinco y devolver cuatro, esto es sumar cuatro. Memorizar estas guías constituirá el secreto para el uso eficiente del ábaco.

Capítulo 2 Análisis y Diseño del Sistema

En el presente capítulo se describe exhaustivamente el problema a resolver, se desarrolla el diagrama de casos de uso para la captura de los requerimientos y se propone la arquitectura modular del agente a desarrollar.

2.1 Descripción del Problema

Como se mencionó anteriormente hoy en día muchas instituciones de Educación Media Superior, e Instituciones Universitarias están empleando los mundos virtuales para impartir cursos a distancia, entre estos mundos virtuales se destaca Second Life. Debido a este crecimiento del uso de mundos virtuales surgen nuevas tecnologías, con la finalidad de mejorar la educación a distancia (SLOODLE [15] fusión de SL y Moodle). En este trabajo se desarrolla un agente de software capaz de controlar a un avatar para que enseñe las operaciones básicas del ábaco a otros agentes o avatares controlados por personas. Para lograr esto, el agente que controla al avatar (denominado Israel), posee cuatro módulos a través de los cuales, logra cumplir su meta (Ver Figura 2.1).

- **Módulo 1 Procesamiento de Lenguaje Natural (PLN)**: Toma todo el texto recibido del mundo virtual y lo procesa, enviando una respuesta adecuada a la situación.

- **Módulo 2 Razonamiento**: A través de este módulo el agente puede determinar el estado del mundo virtual y toma una decisión de qué realizar.

- **Módulo 3 Procesamiento Visual**: Procesa información recibida de dos sensores y así puede determinar la dirección que debe tomar el avatar que está controlando.

- **Módulo 4 Movilidad**: Se apoya del módulo de procesamiento visual, una vez que ya se determinó la dirección que debe tomar el avatar, el agente comienza a desplazar al avatar hasta encontrar al objetivo (otro avatar).

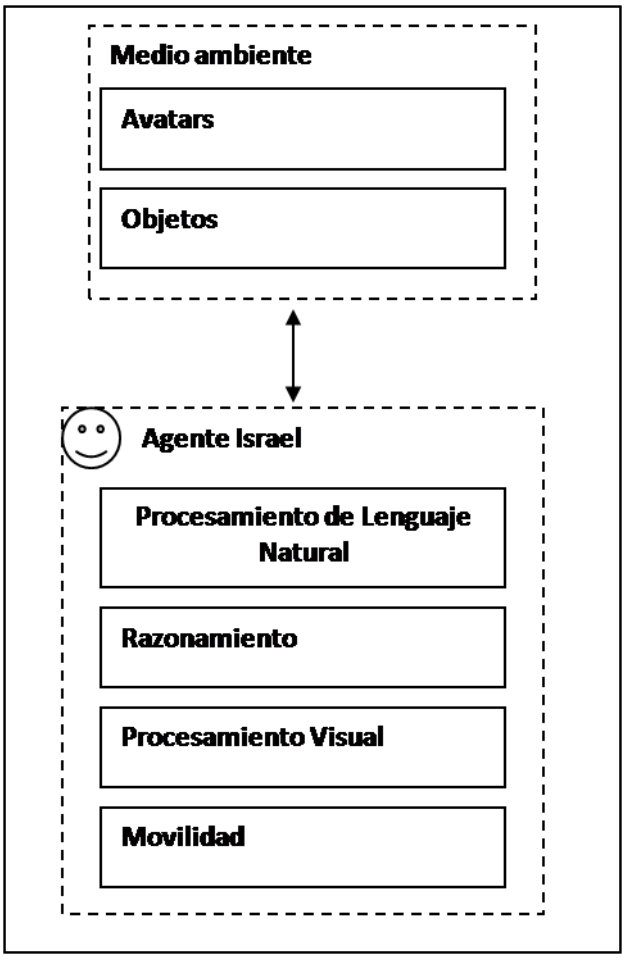


Figura 2.1 Arquitectura del agente Israel.

De acuerdo a todo lo anterior el diagrama de casos de uso el agente contiene (Ver Figura 2.2) tres comportamientos principales:

- Conectarse al mundo virtual, usando por default al avatar Israel.
- Interactuar con objetos dentro del mundo virtual, esto se refiere a como el agente va a hacer uso del ábaco. Para hacerlo se auxilia del envió y recepción de mensajes usando el protocolo EasyTalk, y para comunicarse con algunos objetos utiliza el PLN, ya que es la manera en que algunos objetos interactúan con el usuario; también cabe mencionar que hace uso

del módulo de razonamiento para actuar de manera adecuada con los objetos.

- Interactuar con avatars dentro del mundo virtual. Para que el agente logre interactuar con otros avatars controlados por humanos o agentes, tiene que hacer uso del PLN, pero este módulo hace uso del módulo de movilidad y envió de mensajes; a su vez el módulo de movilidad hace uso del procesamiento de visión, del módulo de sensores y del módulo de envió y recepción de mensajes. Por otra parte el módulo de visión hace uso de los sensores y del módulo de envió y recepción de mensajes. Para mejorar la interacción con otros avatars es necesario usar el módulo de razonamiento.

A partir de las tareas que debe llevar a cabo el agente se desarrolla el diagrama de casos de uso general, que se aprecia en la figura 2.2.

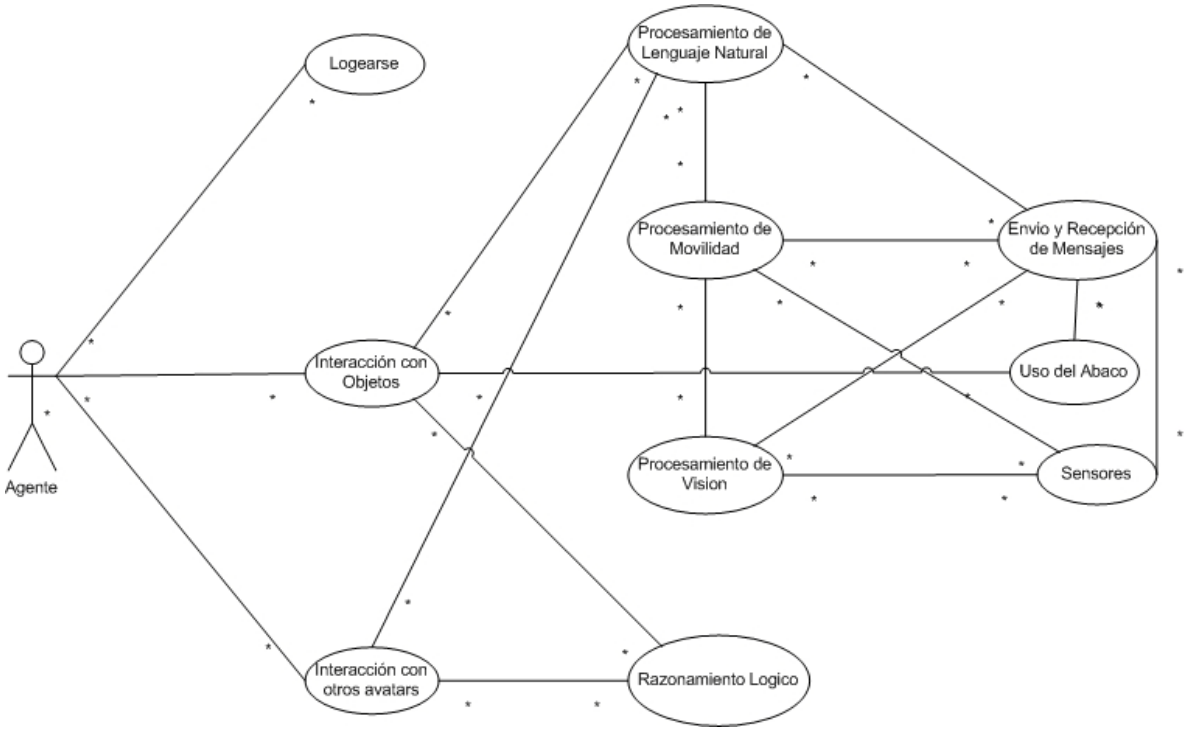


Figura 2.2 Diagrama de casos de uso.

Capítulo 3 Implementación

En este capítulo se discuten los algoritmos empleados para la implementación del agente; además los requerimientos necesarios en hardware y software. Por último se muestran los resultados obtenidos en la implementación del mismo.

3.1 Módulos del Agente

A continuación se muestran los algoritmos empleados en la implementación de los módulos del agente.

3.1.1 Módulo de Procesamiento del Lenguaje Natural (PLN)

Recepción y Envío de mensajes Escritos

El agente se encarga de responder a los mensajes recibidos por otros avatars, para ello el debe ser capaz de identificar con que avatar está conversando. Esto es importante porque el avatar manejado por el agente Israel, puede recibir mensajes no sólo de otros avatars, sino también de objetos. Por todo esto es necesario considerar que para el envío y recepción de mensajes la plataforma maneja canales, que pueden ser públicos y privados, en los cuales se ocupa el protocolo EasyTalk[6]. El protocolo es implementado usando el canal de comunicación (los canales se pueden establecer del 0 hasta el tamaño de un entero de 32 bits), un identificador (conocido como key), un nombre (nombre del avatar) y el mensaje (cadena enviada por un avatar).

Una vez que el agente recuperó el mensaje se va a encargar de procesarlo y una vez procesado dará el mensaje adecuado para responder. Por el momento el agente busca en su base de datos el mensaje con el que responderá de acuerdo al mensaje recibido. La base de datos manejada por el agente posee el siguiente modelo entidad-relación (Ver Figura 3.1).

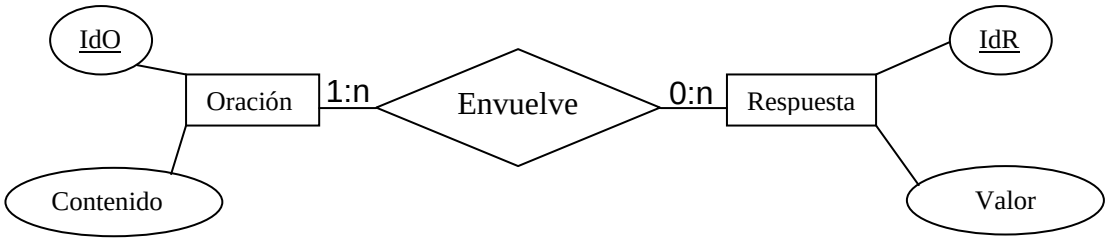


Figura 3.1 Modelo Entidad-relación

La llave primaria está compuesta por IdO e IdR, ambas son referencias a la relación oración y a la relación respuesta.

El agente Israel actualiza la base de datos por medio del algoritmo 1:

```
Algoritmo 1: Actualización de la BD
Step 1: Recibe un mensaje, y extrae los términos que lo conforman.
  String cadena = recv(message)
  Respuesta = For each (Oración.IdO=envuelve.IdO) and
  (envuelve.IdR=respuesta.idR) where  (oración.contenido = cadena).
  If (respuesta <>null) then Terminar.
  Else Goto Step 2.
Step 2: Términos = Procesa(cadena) // enriquecimiento de la base de datos
              bandera = 0;
              For each Oj in Oración do:
                  Jaccard_Sim(Término,Oj)= |O_i ∩ O_j / O_i ∪ O_j|
                  If (Jaccard_Sim(Término,Oj) >= 0.5) then
                  bandera = 1
                  Nueva_respuesta = For each (Oración.IdO= envuelve.IdO)
                  and (envuelve.IDR=respuesta.IDR)
                  where ( Oración.contenido = Oj).
              End For
        If (Nueva_respuesta<> null) then Terminar
        If (bandera ==0) then
        Lista_Sinonimos=busca(WordNet, Oi)
        For each Ok in Lista_Sinonimos do:
              Lista_respuesta = For each (Oración.IdO= envuelve.IdO)
                                and (envuelve.IDR=respuesta.IDR)
                                where ( Oración.contenido = Ok)
                                End For
        End For
        If longitud(Lista_respuesta) >1 then
                 Mayor = 0.5; Término = "";
                 For Each Ti    in Lista_respuesta do:
                   For each Oj in oración do:
                      Jaccard_Sim(Ti,Oj)= |T_i ∩ O_j / T_i ∪ O_j|
                     If (Jaccard_Sim(Ti,Oj) >= Mayor ) then
                         Mayor = Jaccard_Sim(Ti,Oj)
                         Término = Ti
                     End For
                 End For
                 Nueva_respuesta = For each (Oración.IdO= envuelve.IdO)
                                   and (envuelve.IDR=respuesta.IDR)
                                   where ( Oración.contenido = Término)
```

```
       If (Nueva_respuesta <> null) then Terminar
Else Goto Step 5.
End if
End if//bandera =0
Step 5:// búsqueda en la RAE
       For each TMi in Términos
          Categoría = categoría_gramatical(TMi) in Real Academia
                   Española
          Nueva_frase= ∪(TMi), ∀ TMi ∈ {adjetivo, sustantivo,
                 verbo, adverbio, etc.}
       End For
Paso 5.1: Como parte del Paso 5 del Algoritmo 1, se busca con
          Google la definición de la cadena y si la encuentra,
          devuelve como respuesta la definición de la misma. En
          caso contrario, utiliza la nueva frase y realiza una
          búsqueda con Google, si encuentra la definición de la
          nueva frase, la devuelve como respuesta y termina. En
          caso de no encontrarla, intenta buscar frases similares
          a la nueva frase construyendo frases bigramas con los
          términos de la nueva frase. De todas ella selecciona la
          de mayor grado de similitud y superior a 0.5,
          considerando la similitud de Jaccard. Si por último no
          encuentra en Google una respuesta, a partir de las
          definiciones descargadas, construye la misma utilizando
          el Diccionario de la Real Academia Española. Almacena
          en la base de datos la Nueva_respuesta encontrada y la
          nueva frase construida.
```

3.1.2 Módulo de Razonamiento

Para reaccionar a los diversos comportamientos de los avatars, el agente Israel utilizará razonamiento deductivo, utilizando lógica, para codificar una teoría que establezca la mejor acción a tomar dada una situación. Para esto es necesario definir un conjunto de reglas, una base de datos lógica para describir el estado actual del mundo y el conjunto de acciones que el agente puede realizar. Por lo tanto se definen los siguientes predicados (Ver Tabla 3.1):

Tabla 3.1 Tabla de Predicados

Predicado	Descripción
UbicaI(Isla)	El agente se localiza en una (Isla)
Ubica(X,Y,Z)	El agente está ubicado en (X,Y,Z)
Facing(d)	El agente está viendo hacia la dirección (d)
Teach(j)	El agente está enseñando el manejo del ábaco a un avatar (j)
Ocio()	El agente se encuentra en estado ocioso ()
Move(X,Y,Z)	El agente se traslada hacia (X,Y,Z)
MoveI(Isla)	El agente va hacia la (Isla)
Busca(j)	El agente busca un avatar (j)
Identifica(u)	El agente identifica objetos o avatars (u)
RMensaje(m)	El agente recibe mensajes del avatar (m)
EMensaje(n)	El agente envía mensaje al avatar (n)

Posibles acciones que puede realizar el agente Israel:

Ac = {avanzar, retroceder, girar, escribir mensaje, mandar mensaje, teletransportar, interactuar, sumar, restar, multiplicar, dividir }

A continuación se enumeran las reglas que determinan las acciones más relevantes a realizar por parte del agente al manejar el avatar:

Regla 1: UbicaI (I) ^ Ocioso() ^ ¬Busca(A) ^ ¬Identifica(A) > Do(teletransporta).

Regla 2: UbicaI (I) ^ ¬Ocioso() ^ Busca(A) ^ ¬Identifica(A) ^ Move(X,Y,Z) > Do(avanzar).

Regla 3: UbicaI (I) ^ ¬Ocioso() ^ Busca(A) ^ Identifica(O) ^ Move(X,Y,Z) > Do(retroceder).

Regla 4: UbicaI (I) ^ ¬Ocioso() ^ Busca(A) ^ Identifica(O) ^ Move(X,Y,Z) > Do(girar).

Regla 5: UbicaI (I) ^ ¬Ocioso() ^ Identifica(A) ^ RMensaje(M) ^ Facing(D) ^ Ubica(X,Y,Z) > Do(escribe mensaje).

Regla 6: UbicaI (I) ^ ¬Ocioso() ^ Identifica(A) ^ Ubica(X,Y,Z) ^ Teach(Op) ^ EMensaje(MA1) ^ RMensaje(M) > Do(suma).

Regla 7: Ubical (I) ^ ¬Ocioso() ^ Identifica(A) ^ Ubica(X,Y,Z) ^ Teach(Op) ^ EMensaje(MA2) ^ RMensaje(M) > Do(resta).

Regla 8: Ubical (I) ^ ¬Ocioso() ^ Identifica(A) ^ Ubica(X,Y,Z) ^ Teach(Op) ^ EMensaje(MA3) ^ RMensaje(M) > Do(multiplicación).

Regla 9: Ubical (I) ^ ¬Ocioso() ^ Identifica(A) ^ Ubica(X,Y,Z) ^ Teach(Op) ^ EMensaje(MA4) ^ RMensaje(M) > Do(división).

Donde:

- ^ representa un “AND” lógico.
- > representa un “ENTONCES” lógico.
- ¬ representa un “NOT” lógico.
- nombre_función(parámetros_función) representa una función y parametros.

Además,

I = {I1, …. In } islas accesibles

A = {A1, .. , Am} avatars visibles

(X,Y,Z) = {(0,0,0),….,(a,b,c) } coordenadas posibles dentro de una isla

O = {O1, …..,, Ok} objetos accesibles dentro de una isla

M = {M1, ….., Mp} mensajes de canal público

D = {norte, sur, este, oeste, sureste, suroeste, noreste, noroeste}

Op = {suma, resta, multiplicación, división} operaciones posibles sobre el ábaco

MA1 = {aprender a sumar}

MA2 = {aprender a restar}

MA3 = {aprender a multiplicar}

MA4 = {aprender a dividir}

3.1.3 Módulo de Procesamiento Visual

El agente debe ser capaz de identificar a los otros avatars que se encuentren en su entorno, para lograrlo él tiene que hacer un escaneo de la zona en que se encuentra ubicado, con el objetivo de encontrarlos. Para ello se define un campo de visión para el mismo, dentro de su campo de visión él intenta interactuar con otros avatars. Para dar visión al avatar se usan figuras geométricas que sirven de sensores, con el objetivo de que el agente logre detectar objetos y avatars. Un aspecto importante es dotarlo de la posibilidad de distinguir si encontró a un avatar o a un objeto cualquiera de Second Life. En caso de que se haya topado con un avatar, el agente posee la meta de iniciar una conversación, con él, con el objetivo de proponerle la enseñanza de las operaciones sobre el ábaco. Si detecta que lo que encontró fue un objeto seguirá su proceso de búsqueda (ver Figura 3.2), para lo que aplica el algoritmo 2.

```
Algoritmo 2: Vision_Agent()
Bool enseño=false;
Step 1: For each sensor i.  (i = 1,2)
             J=0 , K=0; minute_inicial= Time (Minute);
                  minute_final = minute_inicial + 3;
                  While (!Empty (List. Sensor (i)) and
                        abs(minute_finalminute_inicial<=3)
                   AV [J] = Extract_Avatar (list)
                     Object [K] = Extract_Object(list)
                   enseño=Move(AV[J], Object[K])//aplicando Algoritmo 3
                   J++;  K++;
                     Minute-final = Time(Minute);
                 End while
Step 2: If (!enseño)  Teletransport();
```

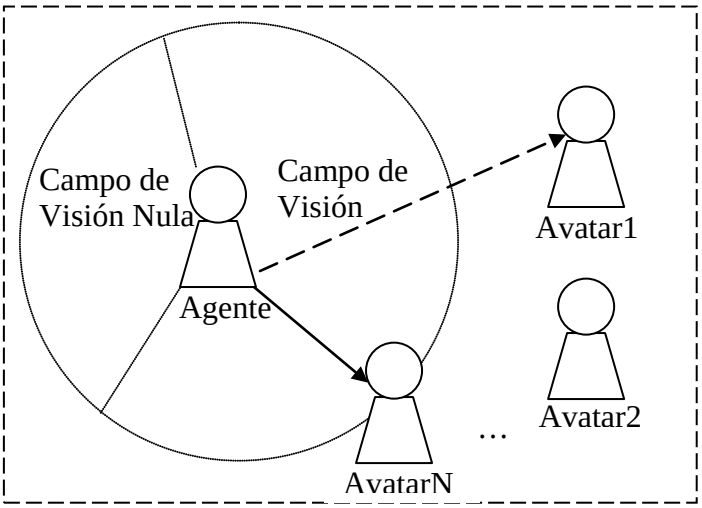


Figura 3.2 Campo de Visión del avatar.

3.1.4 Módulo de Movilidad

El módulo de movilidad es usado en el procesamiento visual, ya que si el agente no encuentra dentro de su campo de visión a un avatar, debe comenzar a trasladarse a otra zona. Primero lo intenta avanzando sobre la misma isla, pero en caso de que no encuentre a ningún avatar el agente tendrá que teletransportarse a otra isla. Hay que destacar que cada isla puede ser un servidor ubicado en algún lugar del mundo (ver Figura 3.3); o en cierto caso será necesario moverse a otro metaverso, para ello se ocupa el Open Grid Protocol (OGP)[8], el cual es de reciente creación (verano 2008) por parte de las empresas IBM y Linden Labs. Con la utilización de este protocolo es posible que un avatar se teletransporte de un metaverso a otro por ejemplo de OpenSim a Second Life.

OGP es un conjunto de protocolos, entre ellos se pueden destacar OGP Authentication y OGP Teleport. El agente para moverse aplica el Algoritmo 3.

```
Algoritmo 3. Move (Avatar, Objeto)
Step  1. variable global coordenada
Step  2. Thread 1 (establece la dirección)
                  While (!Found (Avatar))
                    coordenada = Establece_Dirección(Avatar)
                  End while
                  If(Found(Avatar))
                       Teach Abacus_operations()
                       Return true
                   Else
                       Return False
         Thread 2:  If (verifica(coordenada))   walk(avatar, coordenada,
adelante)
         Thread 3:  If (verifica (coordenada)) walk(avatar, coordenada,
atrás)
         Thread 4  If (verifica (coordenada)) walk(avatar, coordenada,
derecha)
         Thread 5  If (verifica (coordenada)) walk(avatar, coordenada,
izquierda)
         Thread 6  If (verifica (coordenada)) walk(avatar, coordenada,
giro_derecha)
         Thread 7  If (verifica (coordenada)) walk(avatar, coordenada,
giro_izquierda)
```

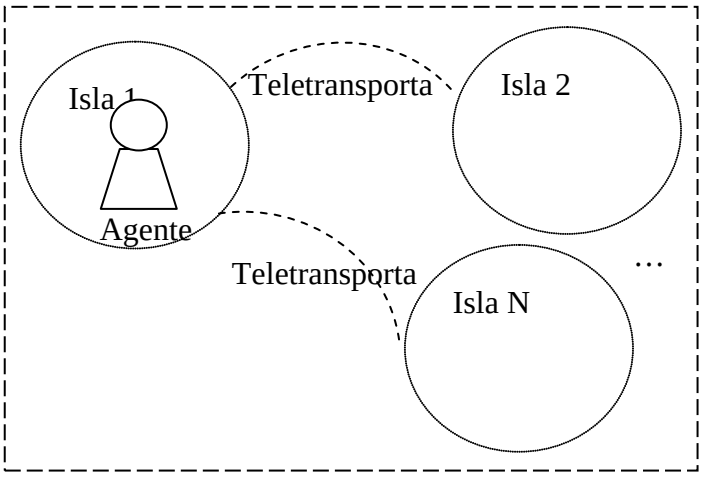


Figura 3.3 Teletransportación del avatar en distintas islas sin restricción.

Con el análisis y diseño previo del agente se pueden distinguir cuatro clases importantes, que dan como resultado el diagrama general de clases (ver figura 3.4).

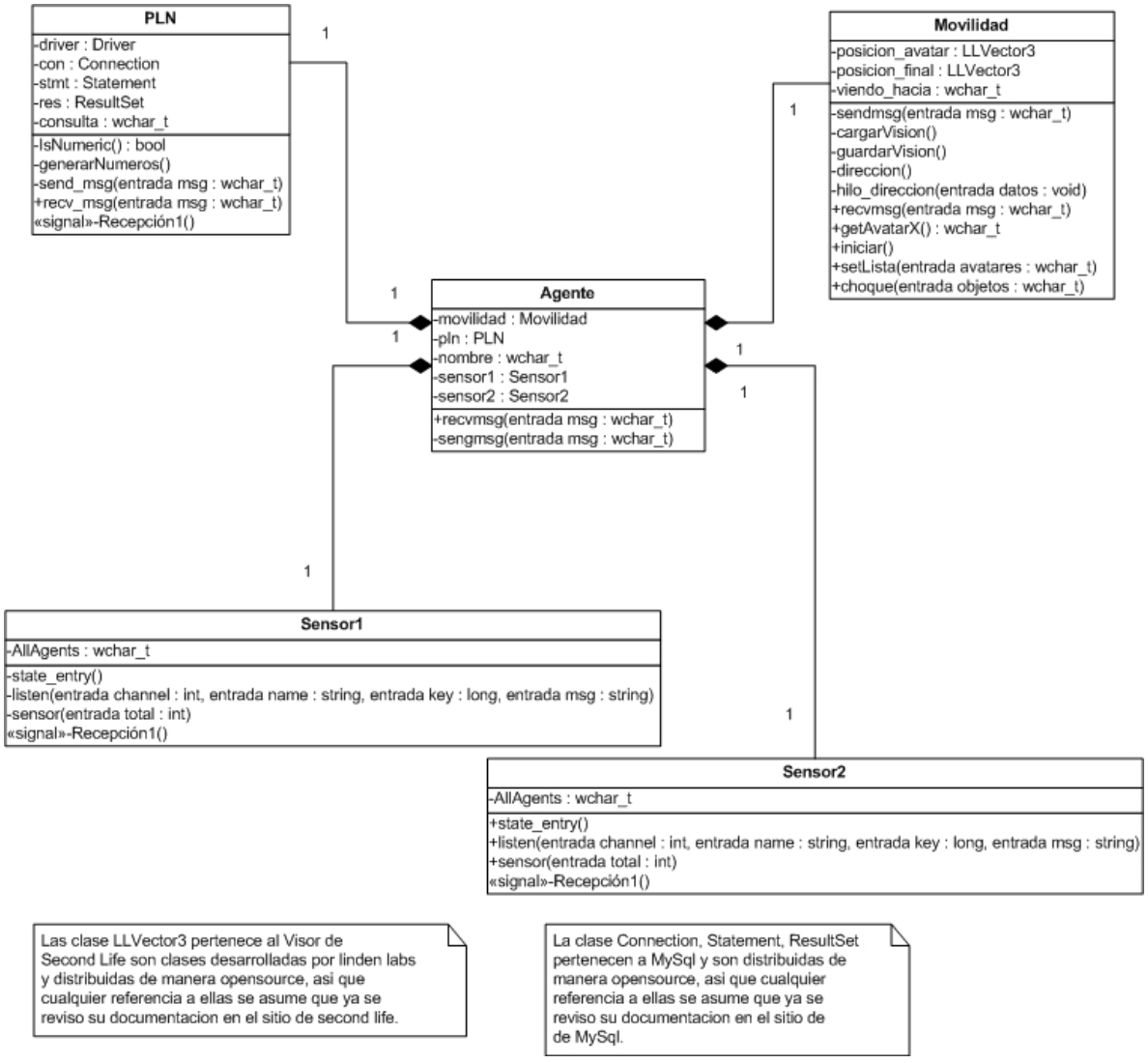


Figura 3.4 Diagrama general de clases

3.1.5 Algoritmo para manejar el ábaco

Para entender el Algoritmo 4, es necesario saber que el ábaco se divide en dos secciones. Se tiene una sección que solamente posee dos bolas, denominadas altobolas, por otra parte se tiene otra sección que posee como máximo 5 bolas, denominadas por su parte hipobolas.

Es importante destacar que el ábaco posee 49 bolas, distribuidas en 7 columnas. Cada hipobola en la primera columna tiene un valor de 0.01, y cada altobola un

valor de 0.05, las segunda columna representa al 0.1 para las hipobolas y 0.5 para las altobolas (para mayor entendimiento revisar la sección 1.4.2).

En la sección 1.4.3 que corresponde a la explicación de cómo se hacen las sumas en el ábaco se exponen 17 reglas para llevar a cabo una suma, pero estas reglas son insuficientes porque si uno desea utilizar el máximo número en el ábaco (166,666.65) esto no es posible, por eso algunas de las reglas se modifican y otras se dejan intactas y las reglas que se añaden hacen uso de la altobola del extremo, que en las 17 reglas se dice que no se use; además se hace uso de la hipobola del extremo, que en las 17 reglas se dice que no se use. El mismo problema se presenta al realizar restas, multiplicaciones y divisiones así que se hacen cambios similares a sus reglas.

La tercera columna representa a la unidad, y así sucesivamente ambos tipos de bolas incrementan su valor multiplicando por 10. El algoritmo aplicado por el agente es el algoritmo 4.

Algoritmo 4.

```
Step 1: Al inicio el agente establece la cantidad representada en el
ábaco a 0.0
Step 2: El agente recibe y recupera mensaje(mensaje,Tipo).
            If (Tipo =  ábaco)
                  Recupera valor (bola)
                  Actualiza las variables de entorno
            Else
               Utilizando el módulo de procesamiento de lenguaje
               Natural, analiza mensaje
               Determina tipo de operación (suma, resta, multiplicación,
               división ó representación)
               If (tipo_de _operación = representación) extrae del
               mensaje el número para representarlo, posterior  resetea
               al ábaco.
               Else representa la operación, mediante el movimiento de
               las bolas y sigue las reglas del ábaco.
```

Algo importante a destacar es que la primera vez que el agente detecta que se va a llevar a cabo una operación, es decir, suma, resta, multiplicación o división, el agente se prepara para recuperar el/los operando(s) y el/los operador(es); el operando puede ser dado en forma numérica o forma escrita (Ejemplo Forma numérica=5, Forma Escrita=Cinco). Para que el agente pueda interpretar la manera escrita se genera una tabla, esta tabla sólo se genera una vez y después

se mantiene durante todo el ciclo de vida del agente; en esta tabla se almacenan los nombres principales de los números (ver Tabla 3.2), y después se generan todos los demás nombres que no son principales; esto es porque al escribir un número en letra por lo general estos se componen de la escritura anterior de los otros números, así teniendo los nombres principales a partir de ellos se pueden generar todos los demás nombres; obviamente esta tabla es finita y por lo tanto sólo se puede generar la escritura hasta cierto número, en este caso el agente puede interpretar la manera escrita del número desde el 0 hasta el 999,999,999.99 sin importar si es positivo o negativo. Una vez que ya se tiene la tabla generada el agente realiza una búsqueda secuencial. Para encontrar el número en la tabla una vez encontrado recupera los índices y es así como él realiza la conversión del número escrito a número en dígitos.

El agente puede realizar N operaciones y por lo tanto este procedimiento puede ser largo, dependiendo de lo que se tenga que hacer; es por eso que la tabla sólo se genera una vez, y se mantiene en memoria.

Tabla 3.2 Se muestran los números principales

Número	Cantidad Escrita
1	Uno
2	Dos
3	Tres
4	Cuatro
5	Cinco
6	Seis
7	Siete
8	Ocho
9	Nueve
10	Diez
11	Once
12	Doce
13	Trece
14	Catorce
15	Quince
16	Dieciséis
17	Diecisiete
18	Dieciocho
19	Diecinueve

20	Veinte
21-29	“Veinte y”+Generados por el agente
30	Treinta
31-39	“Treinta y”+Generados por el agente
40	Cuarenta
41-49	“Cuarenta y”+Generados por el agente
50	Cincuenta
51-59	“Cincuenta y”+Generados por el agente
60	Sesenta
61-69	“Sesenta y”+Generados por el agente
70	Setenta
71-79	“Setenta y”+Generados por el agente
80	Ochenta
81-89	“Ochenta y”+Generados por el agente
90	Noventa
91-99	“noventa y”+generados por el agente
100	Cien
101-199	“ciento”+generados por el agente
200-999	Generado+“cientos”+generado por el agente
1000	Mil
1001-999999	Generado+“mil”+generado por el agente
1000000	Un millón
1000001-1999999	“Un millón”+generado por el agente
2000000-999999999	Generado+”millones”+generado por el agente

3.1.6 Algoritmo General

A partir de las acciones que el agente debe seguir por medio de sus 4 módulos y del algoritmo de manejo del ábaco se formula lo que se ha denominado Algoritmo 5.

Algoritmo 5:
```
Step 1: Logearse al mundo virtual, usando por default al avatar Israel.
Step 2: Aplicar Algoritmo 2.
Step 3. Después de un tiempo  cerrar aplicación.
```

En el Algoritmo 5 sólo se realiza un llamado al algoritmo 2, ya que éste desencadena la ejecución total de la aplicación.

3.2 Instalación y Configuración de las Herramientas

En el apéndice A se encuentra todo lo relacionado con la instalación y configuración de las herramientas del Visor de Second Life (Ver Sección 1.3) [20].

3.3 Hardware Necesario para la Implementación

La implementación y pruebas realizadas del agente se llevaron a cabo sobre el siguiente modelo de computadora (HP-Pavilion pa6280la):

- Procesador i5-750 a 2.66 Ghz (procesador de cuatro núcleos, cada núcleo puede ejecutar una tarea).
- Memoria RAM de 6 GB DD3.
- Un disco duro de 750 GB debido a que la BD del agente crece a un ritmo muy rápido.
- Una tarjeta de gráficos de 1 Giga de video dedicado NVidia modelo GT 220 (lamentablemente el visor de Second Life requiere de muchos recursos una vez que tiene al agente funcionando).
- Sistema Operativo Windows Seven Home Premium 64 bits.

Para la realización de pruebas se instaló un servidor OpenSim, sobre el siguiente modelo de computadora (GatewayGT 3238m):

- Procesador Core 2 Duo a 1.86 Ghz (procesador de dos núcleos cada núcleo puede ejecutar una tarea).
- Memoria RAM de 2 GB DDR2.
- Un disco duro de 400 GB debido a que la BD de OpenSim crece a un ritmo muy rápido.
- Una tarjeta de gráficos Intel G965 de 128 Megas de video dedicado (se utiliza esta tarjeta para correr a otro avatar dentro de OpenSim).

- Sistema Operativo Windows Vista Home Premium 32 bits.

El utilizar un equipo de menor potencia que los antes mencionados ocasiona que el agente no funcione de manera adecuada, se recomienda el uso de estos equipos o de equipos superiores.

NOTA 1: Se recomienda la instalación de una red local sólo para pruebas de depuración, para conectarse a Second Life se recomienda una conexión DSL de al menos 1Mbit/segundo.

3.4 Desarrollo de la Aplicación

El lenguaje usado es Visual C++ 2008 de la versión Express de Microsoft, se eligió este lenguaje debido a que es el mismo lenguaje ocupado para compilar el Visor de Second Life y el agente va a utilizar este visor para conectarse al mundo virtual. La versión utilizada del visor es la 1-21-r99587, se eligió esta versión debido a que cuando se comenzó la tesis era la versión estable del visor hoy en día hay versiones más actuales y no hay ningún problema con actualizar la versión del visor para el agente, el módulo de procesamiento de lenguaje natural usa una base de datos para almacenar todo el lenguaje posible.

NOTA: Los sensores deben encontrarse dentro del mundo virtual por lo tanto están programados en el lenguaje LSL.

3.4.1 Implementación del Módulo de Procesamiento de Lenguaje Natural

Para la implementación de este módulo se creó una clase que se encarga de la conexión a la BD y que tiene por métodos principales el enviar y recibir mensajes en el mundo virtual.

Para la recepción de cada mensaje se crea un hilo, este hilo está activo mientras se responde el mensaje recibido, se pueden tener N cantidad de hilos activos, esta es una de las razones principales por las cuales la computadora tiene que ser potente (ver sección 3.2).

Mientras un hilo está activo intentando responder un mensaje del mundo virtual, otro hilo puede crearse intentar responder el mensaje recibido. Se decidió este mecanismo de comunicación, porque un mensaje puede requerir mucho tiempo en ser atendido mientras que otro mensaje puede ser rápidamente atendido, además la llegada de cada mensaje es de manera asíncrona, así que es más conveniente crear un hilo por cada mensaje recibido.

Este módulo se auxilia de eSpeak 1.4.3 para hacer la conversión de texto a voz (sólo en idioma español), así se le dota al agente de una voz; y utiliza solicitudgetCLR 1.0.0 para realizar las solicitudes http (conexión a google y la DRAE). Como se mencionó en la sección 3.2 el agente necesita una computadora con un disco duro de gran capacidad debido a que la aplicación solicitudgetCLR 1.0.0 descarga a disco duro la información de la RAE y de Google; cuando ésta información está en el disco duro el agente detecta y recupera esa información y la comienza a procesar de manera adecuada. Toda la información descargada nunca es eliminada por el agente, esto sirve para posteriores análisis y mejoras en cuanto a como el agente procesa la información.

3.4.2 Implementación del Módulo de Movilidad

Para este módulo se creó una clase la cual tiene en ejecución varios hilos cada hilo se encarga de una dirección (derecha, izquierda, adelante, atrás, giro_izquierda, giro_derecha), y además consta de un hilo que se ejecuta de manera infinita, este hilo es encargado de activar o desactivar los otros hilos de acuerdo a los mensajes recibidos por los sensores; de acuerdo a la información recibida por los mismos se define la dirección del avatar.

3.4.3 Implementación del Módulo de Visión

El módulo de visión consta de 2 sensores, los cuales están implementados dentro del mundo virtual, se ejecutan casi de manera indefinida pero se alteran sus estados, con el objetivo de que no se estén ejecutando todo el tiempo, debido a que en ciertos intervalos de tiempo sus mensajes no son importantes, debido a que el agente no necesita esa información en ese momento, por lo tanto se

desactivan y activan a conveniencia, eso lo decide el agente. Ejemplo de cuando se desactivan, cuando el agente esta interactuando con otro avatar u otro agente.

3.4.4 Implementación del Ábaco

La implementación del ábaco se llevó a cabo dentro del mundo virtual, este sólo admite dos decimales, y está programado para comportarse como lo haría un ábaco en la vida real. El ábaco utiliza EasyTalk para la comunicación con el agente. El número más grande representado en el ábaco es el 166666.65.

3.5 Pruebas

Como ya se mencionó para las pruebas de depuración se instaló un servidor OpenSim (Ver Apéndice B) debido a su similitud con Second Life, para las pruebas finales se ejecutó el agente dentro de Second Life teniendo los mismos resultados obtenidos dentro de OpenSim.

En una computadora se instaló el servidor de OpenSim y se configuró todo lo necesario para su correcto funcionamiento (Ver Apéndice B).

En otra computadora se hizo funcionar otro visor de Second Life controlado por un humano, para llevar a cabo pruebas locales. Mientras que el agente estuvo funcionando en otra computadora.

3.6 Resultados y Ejemplos

A continuación sólo se describen los puntos más importantes de las pruebas realizadas; todo lo que no se menciona es porque funcionó de manera correcta y óptima, además ha sido descrito en el análisis y diseño.

3.6.1 Visibilidad y Movilidad

Con respecto a la visibilidad se envían mensajes entre el agente y los sensores, la llegada de estos mensajes es de manera asíncrona; hay dos factores que influyen en esta comunicación, primero dependen de la velocidad de la conexión a Internet y además dependen de la velocidad del servidor para ejecutar los scripts. En las pruebas el agente tuvo un rendimiento medio debido a los dos

factores previamente mencionados, teniendo un leve retraso en la actualización de los sensores; lo cual provoca que haya un leve desplazamiento entre los avatars (Ver Figura 3.5). Otro factor importante que influye es el tiempo que le lleva al agente procesar la información recibida por los sensores, ya que los sensores envían un mensaje con determinado formato que después será procesado por el agente. El sensor que detecta a los avatars envía un mensaje con el siguiente formato “X Y Z | NombreAvatar1 X Y Z | NombreAvatar2 X Y Z | … | NombreAvatarN X Y Z” como se puede apreciar es una lista de nombres de avatars y sus posiciones, excepto el primer elemento no contiene nombre, ya que es la posición actual del avatar Israel como delimitador se tiene a “|” entre cada avatar. El sensor responsable de detectar objetos envía una lista de los objetos que se detectaron en un radio de un metro con respecto al avatar Israel. Una vez que el agente recupera esa información de los sensores es pasada al módulo respectivo y esto desencadena una serie de acciones como son desviación de la ruta del avatar para esquivar los obstáculos. Algo muy importante a mencionar es que estos objetos que son un obstáculo pueden encontrarse en movimiento a determinada velocidad o con velocidad variable o simplemente estáticos, es decir, sin movimiento. Esto sucede porque dentro del mundo virtual se pueden construir motocicletas, autos, aviones, pelotas, perros, gatos, etc., es decir, que están en movimiento.

Figura 3.5. En la figura se puede observar al avatar Israel (avatar con forma de hombre) controlado por el agente. El agente está intentando posicionar al avatar Israel cerca del avatar Darnes.

3.6.2 Procesamiento de Lenguaje Natural (PLN)

Para atender cada mensaje del mundo virtual se crea un hilo, así se crean N hilos para atender los mensajes recibidos por objetos y por los avatars. La creación de estos hilos es de manera dinámica y el número de hilos activos depende mucho de la potencia de la computadora que se esté usando para la ejecución del agente. Por lo tanto el agente puede responder a múltiples mensajes al mismo tiempo, pero no necesariamente en el orden en que fueron recibidos. Ya que cada hilo consume una cantidad de tiempo diferente dependiendo de la información que debe procesar (Ver Figura 3.6).

Aunque el agente consume mucho tiempo para conectarse a Internet, los resultados son satisfactorios. Ejemplo de esto es que el agente puede responder a preguntas como "de que color es el caballo blanco de Napoleón" la respuesta que

da es "pues na guita es blanco son preguntas de agilidad mental y capciosas", como se puede observar la respuesta más correcta seria "es de color blanco", sin embargo la respuesta que da el agente fue la respuesta encontrada en Internet utilizando Google, y ésta es una muy buena aproximación a la respuesta real.

A pesar de que se usa eSpeak para dotar de voz al agente, aun tiene errores de pronunciación, pero estos pueden ser arreglados mejorando eSpeak ya que es de código libre.

Figura 3.6 En la figura se puede observar al avatar Israel (avatar con forma de hombre) controlado por el agente. El agente responde a una pregunta hecha por el avatar Darnes, el avatar Darnes que es controlado por un humano.

Algo importante es que el visor de Second Life hace revisiones sobre el estado del visor del agente para saber si hay una persona manejándolo y evitar que este sea expulsado. El agente para engañar las revisiones del visor de Second Life debe enviar de vez en cuando pulsos de teclas que activen eventos que producirán

animaciones las cuales están asociadas a una tecla de función en el teclado, de esta manera se engaña al visor al momento de hacer las revisiones. Por ejemplo:

1: Cuando el agente manda el mensaje de “hola” en ese momento se desencadena una animación que hace al avatar mover un brazo en señal de saludo.

2: Cuando el agente va a representar algún número sobre el ábaco este desencadena una animación del agente apuntando hacia el ábaco.

3: Cuando el agente está armando un mensaje activa una animación del avatar “escribiendo” y una vez enviado el mensaje se detiene la animación.

4: Cuando el agente esta accediendo a Internet hace una animación de pensando.

5: Cuando el agente está realizando una suma y va a representar el número en el ábaco se realiza una animación de señalamiento.

6: Cuando el agente no puede responder una pregunta hace una animación de frustración.

Todas esas animaciones son realizadas utilizando diferentes aplicaciones, entre las que destacan dos que son Avimator (Ver Figura 3.7) y QAvimator (Ver Figura 3.8) ambas aplicaciones fueron utilizadas para realizar las animaciones.

La principal diferencia entre Avimator y QAvimator, es que QAvimator permite un mayor control en los movimientos del avatar, mientras que Avimator tiene menos control en el movimiento del avatar.

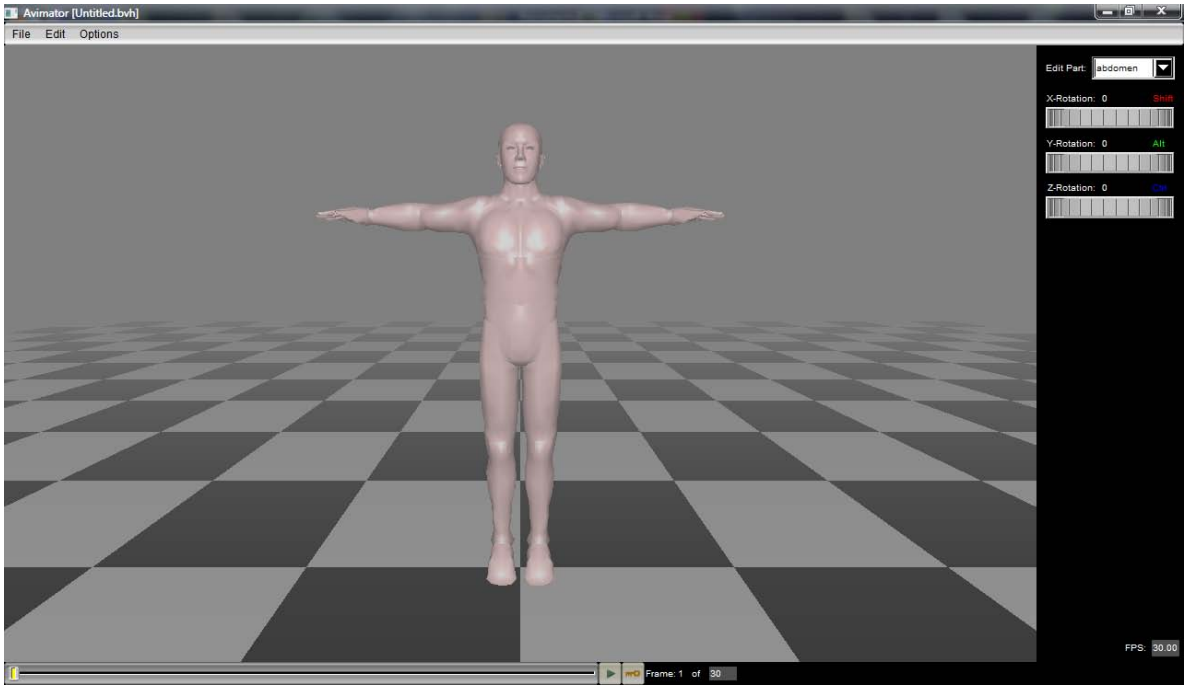


Figura 3.7 Avatar Dummy que muestra como se realiza la animación, se traslada o rota alguna de las partes del avatar como son el brazo, abdomen, piernas y cabeza.

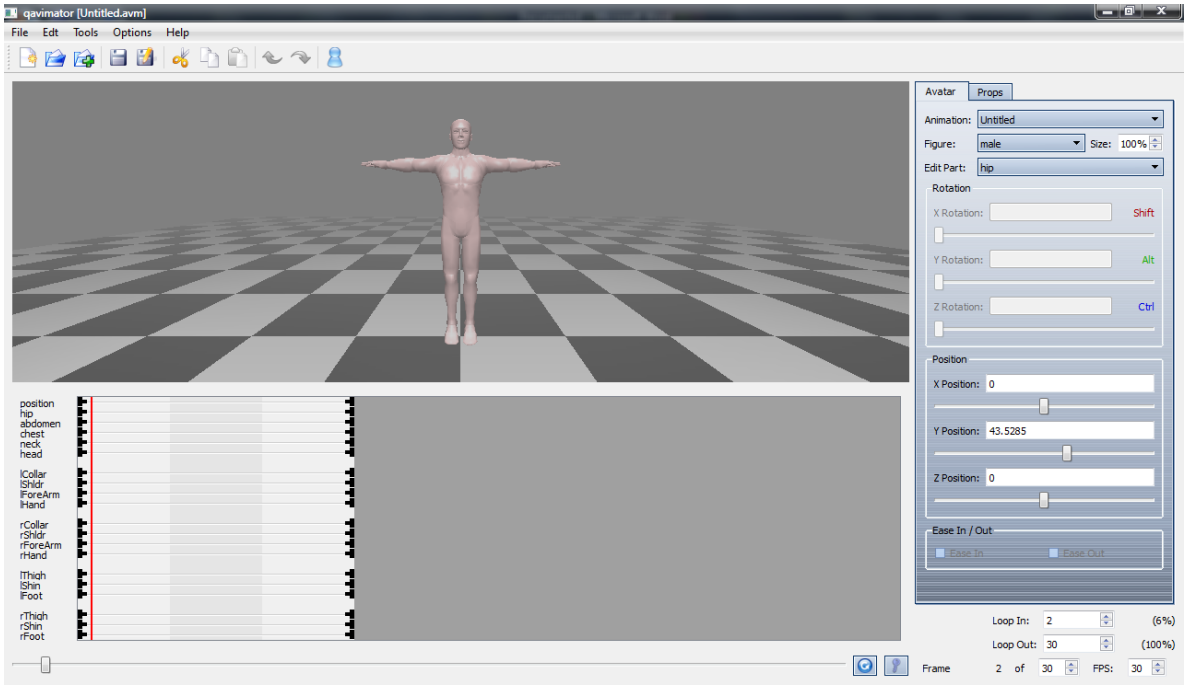


Figura 3.8 En QAvimator se tiene un mayor control sobre al movimiento del avatar.

3.6.3 Uso del Abaco

Para el uso del ábaco se utiliza el protocolo EasyTalk el agente puede saber el estado del ábaco(ver Figura 3.9), pero el inconveniente es el tiempo que lleva actualizar ese estado, debido a que depende de dos factores uno de la velocidad de conexión a Internet y el otro es el tiempo que tarda en ejecutarse los scripts del ábaco, este tiempo depende totalmente del servidor de Second Life y no puede ser medible. Sin embargo el agente logra actualizar el estado del ábaco, pero debido a este retraso, no controlable por él, en determinadas ocasiones puede interpretar mal el estado del ábaco.

El agente puede atender múltiples mensajes del ábaco al mismo tiempo, esto es actualizar el estado del ábaco.

Figura 3.9 En la figura se puede observar al avatar Israel (avatar con forma de hombre) controlado por el agente. El agente está representando un número sobre el ábaco que es el número cinco.

Conclusiones y Trabajo Futuro

Los objetivos propuestos en la tesis fueron cumplidos de manera satisfactoria, con las pruebas realizadas (ver sección 3.5) se constató que la implementación (ver sección 3.1) del agente cumplió con las expectativas para llevar a cabo sus objetivos.

Se disponía de cierta experiencia en el desarrollo de agentes sobre la plataforma JADE [21], sin embargo el desarrollo de agentes dentro del mundo virtual es totalmente diferente, pues es necesario auxiliarse de sensores y una gran parte de la eficiencia del sistema reside en el envío y recepción de los mensajes y de los cambios continuos que se efectúan dentro del mundo virtual. Otro aspecto importante a considerar es la cantidad de recursos computacionales que consume una aplicación de este tipo.

Se le agregó al módulo de Procesamiento de lenguaje natural la búsqueda a Internet en aras de brindar mejores resultados, sin embargo el tiempo de respuesta del agente se vio afectado sustancialmente. Se tienen 3 versiones diferentes del agente en cuanto a la forma en que ha sido implementado el módulo de procesamiento del lenguaje natural. La versión 1, que es más estable es la que se ha descrito en el presente trabajo. Sin embargo las otras han servido como punto de comparación. La que se denomina versión 2, maneja Gecko para las conexiones que realiza el agente a Internet, cuando debe ofrecer una respuesta solicitada por un avatar manejado por humanos. Esto es mucho más confiable, pero exige mayor cantidad de recursos computacionales. La versión 3 ocupa DLL para la conexión a la Base de datos, lo que provoca que se consuma más tiempo de ejecución y mayor cantidad de recursos.

Como trabajo futuro se propone sustituir el sistema de visión actual que está basado en sensores por un sistema de visión 3D, con esto la teletransportación del agente se ve beneficiada al no depender de objetos programados en LSL y así se pueda teletransportar entre metaversos utilizando el protocolo OGP.

Se propone mejorar el módulo de procesamiento de lenguaje natural, agregando análisis de contexto, esto sin duda hará que el agente tarde más tiempo en responder, pero mejorarían los resultados en la respuesta. Además consideramos que se puede mejorar el tiempo de respuesta utilizando la tecnología CUDA[13] de NVIDIA, para cálculo paralelo, ya que el visor de Second Life no hace uso de la misma. En este módulo se podría agregar soporte a múltiples idiomas esto se lograría usando la aplicación solicitudgetCLR, además el agente se debe conectar al traductor de Google, y así realizar la traducción de las frases recibidas a un determinado idioma. También se propone mejorar la aplicación solicitudgetCLR (encargada de conexiones a Google y a la RAE) para efectuar un mejor análisis de páginas web, o la otra opción es sustituir completamente esa parte por Gecko[14]. Sin duda Gecko sería la mejor opción, pero se necesitan una mayor cantidad de recursos computacionales (versión 2 del agente).

El agente tiene ya una voz propia pero aun no puede analizar audio, esto se puede lograr utilizando eSpeak pero haciendo mejoras a esta aplicación para que pueda ser usada de manera adecuada por el agente, además se debe tener en cuenta que para lograr este propósito es necesario instalar además de OpenSim un servidor de Asterisk[12] e instalar módulos adicionales a OpenSim, para que funcione el audio en el mundo virtual, y diseñar e implementar los algoritmos correspondientes.

Por último, se puede mejorar los algoritmos utilizados en la movilidad para que esta sea más estética, es decir, parezca un poco más humana y no se vea tan robotizada. La movilidad que realiza el agente sobre el avatar es que sólo se puede desplazar sobre los ejes X e Y; como es un mundo tridimensional tenemos a la altura Z, es decir, el avatar puede volar pero en este momento el agente no puede hacer que el avatar vuele debido a que el algoritmo empleado para la movilidad sólo soporta los ejes X e Y; por lo que otra mejora seria dotar al agente de algoritmos de movilidad que soporten el vuelo, es decir, desplazamientos sobre los ejes X, Y y Z.

Referencias

1. Definición de Metaverse, http://en.wikipedia.org/wiki/Metaverse, 2010
2. Reyes-Farfán, N., Sánchez, J. A. 2003. Personal spaces in the context of OAI. Proceedings of the Joint Conference on Digital Libraries (JCDL 2003, Houston, Tex., May)
3. Bringsjord S., Shilliday A., Clark M., Werner D., Taylor J., Bringsjord A., Charpentier E. (2008). Toward Cognitively Robust Synthetic Characters in Digital Environments. Artificial General Intelligence 2008 Proceedings of the First AGI Conference. 171
4. Mundo Virtual Second Life, http://www.secondlife.com
5. Conference on Artificial General Intelligence. http://www.agi-08.org/, 2010
6. Protocol EasyTalk, http://wiki.secondlife.com/wiki/LSL_Protocol/EasyTalk, 2010
7. Speech Synthesizer, http://espeak.sourceforge.net/, 2010
8. Protocol OGP, http://wiki.secondlife.com/wiki/OGP_Base, 2010
9. What is OpenSim?, www.opensimulator.org, 2010
10. Linden Scripting Language, http://wiki.secondlife.com/wiki/LSL_Portal, 2010
11. Institutes in Second Life, http://secondlifegrid.net/casestudies, 2010
12. Voice over Internet Protocols, http://www.asterisk.org/, 2010
13. Cálculo Paralelo de NVIDIA utilizando la tecnología CUDA, http://www.nvidia.es/object/cuda_home_new_es.html, 2010
14. Motor de Navegación para Firefox, http://developer.mozilla.org/es/Gecko, 2010.
15. Fusión de Second Life y Moodle, http://www.sloodle.org/moodle/, 2010
16. Ranking de mundos virtuales, http://gigaom.com/2007/06/13/top-ten-most-popular-mmos/, 2010
17. Conceptos de Second Life, http://es.wikipedia.org/wiki/Second_Life, 2010
18. Instituciones educativas, http://education.secondlife.com/, 2010
19. Conferencia para educación de idiomas en mundos virtuales, http://www.slanguages.net/, 2010
20. Herramientas para compilar en Windows el visor de Second Life, http://wiki.secondlife.com/wiki/Microsoft_Windows_Builds, 2010
21. Framework para desarrollo de Agentes en JAVA, http://jade.tilab.com/, 2010

Apéndice A

Este apéndice trata en como instalar y como configurar las herramientas necesarias para compilar el Visor de Second Life.

Microsoft Windows Builds

From Second Life Wiki

On Windows, there are several options on build (compile) environment of the Second Life.
This page explains how you can compile the Viewer on Microsoft Windows.
Currently, only 32 bit binary is tested. There seems to be several trials to produce 64 bit Windows .EXE of the Viewer. If you did, please write your experience on the discussion page (regardless it was successful or not!)

Contents

Choosing and preparing a compiler

Linden-supported compilers

Supported compiler: **Visual Studio .NET 2005 Professional**
You need to setup the compiler and Microsoft Development tools as follows:

- Setup Microsoft Visual Studio

Community experimental compilers

If you don't have Visual Studio .NET 2005 Professional, you may wish to try one of the following alternatives.

- Visual C++ 2005 Express (instructions, but the screenshots for VS2008 are worth a glance too)
- Visual Studio 2008 (instructions)
- Visual C++ 2008 Express (instructions)

Warning!

Boost support with Visual Studio 2008 is problematic as of this writing. Check VWR-9541[c] before continuing on this path.

NOTE: Make sure you install to paths without spaces in it.

Getting other development tools

You will need to install the following tools to compile the Viewer:

- **CMake** (download CMake)
  - As of this writing, the latest version is 2.6.2. **Note**: There are many known issues with CMake 2.6.0 and 2.6.1 in conjunction with building the Second Life Viewer. CMake 2.4.8 is supported for compiling the 1.21 version of the Second Life Viewer, but 2.6.2 is likely to become the new minimum requirement in the near future.
- **Cygwin** (download Cygwin)
  - When you run the cygwin setup utility make sure you have selected to install **patchutils**, **flex**, and **bison** (all located under "devel") which are not part of the default install. (If you missed one of these, the easiest thing to do is to re-run the entire installation.)
- **Python** (download either Python.org Standard Python or ActivePython
  - If you are using a version of Python newer than v2.5, you may need to change the Python.cmake file. See the CMake discussion for details (this change was necessary as of 1.21-r99587 source branch).)
- **The Windows Platform SDK**
  - Get the latest version (as of 23 March 2010) here: Windows SDK for Windows Server 2008 and .NET Framework 3.5 SP1
- **DirectX 9.0 SDK**
  - Get the latest version (as of 29 Oct 2008) here: DirectX 9.0 SDK (August 2008)

Verify that Cygwin, CMake, and Python are in the windows "PATH".
NOTE: DO NOT use the Cygwin version of CMake or Python. The Build will fail. (CMake specifically excludes the Cygwin version of Python, in the 'Python.cmake' file)

Downloading source code

You can download the Viewer source code on the source downloads page. You can also use a version control repository. If you're just starting out, it's probably best to get the latest Release version, rather than a Release Candidate, because the Release Candidates get updated quite often. But if you would rather work with the latest code, go for the version control repository "trunk".
If you're downloading from the source downloads page, there are three packages to get: the source package, the artwork package, and the library package. In versions 1.20 and earlier, Linden packaged the library binaries in the Libs package. For 1.21 and beyond, the CMake develop.py script now downloads **most** of the libraries that were previously in the libs zip file. This saves developers who are tracking trunk from constantly downloading them every update and only downloads updated libraries. As of this writing, there are some pieces packages that still require downloading, so be sure to grab the library and artwork bundles from the source downloads page.
WARNING:

- If the directory path you keep the SL source in has a space in it, the batch file that copies message_template.msg will fail. So, if you unzip or checkout the source tree into, e.g., "C:\Projects\Dir with space in name\Etc\linden", it won't work!
- You should also avoid using non-ASCII (national) characters in the paths, although some localized versions of the tool puts some as a default...
- Unzip or checkout your source tree into a directory that has as short full pathname as possible, since long paths cause some unexpected trouble during the build.

In other words, the easiest way to get this working is to get **source**, **artwork**, and **libs** from the source downloads page and unpack them all into the same directory/folder, which ideally would be a folder in (or near) the root directory with a short name like **sl_1_21_6**.

Installing libraries

SL Viewer depends on some third party libraries. Some of them are open source, some others are not.

Open source libraries

As of Viewer version 1.21, all open source libraries are automatically downloaded as part of the build process, unless you choose to configure a standalone build.

Proprietary libraries

Linden does not include the following proprietary libraries. You will need to follow the instructions to acquire below and copy them to the source path.
However, it probably is a good idea to build an empty directory tree for the files below and first copy the files there and once completed, copy the whole tree to the actual source folder (like *XCOPY OLIB SL_1_16_0_5 /S*). The reason is, that these

steps are cumbersome and will have to be repeated for each new release (at least if you keep the source for each release in it's own folder). If you do not want to do this, of course you can just copy the files directly into the linden source paths.

```
rem OLIBS.CMD to build a folder tree for 3rd party libraries and includes
md olibs
md olibs\linden\
md olibs\linden\libraries
md olibs\linden\libraries\include
md olibs\linden\libraries\i686-win32
md olibs\linden\libraries\i686-win32\lib_release
md olibs\linden\libraries\i686-win32\lib_debug
md olibs\linden\libraries\i686-win32\include
md olibs\linden\libraries\i686-win32\include\GL
md olibs\linden\libraries\i686-win32\include\quicktime
md olibs\linden\indra
md olibs\linden\indra\newview
```

1.5.1. Fmod

- Download & extract fmod 3.75 api for win32, under section heading **FMOD 3 Programmers API** (later versions, like FMOD Ex, are incompatible).
- Copy "fmodapi375win\api\inc\fmod.h" to "linden\libraries\include"
- Copy "fmodapi375win\api\inc\fmod_errors.h" to "linden\libraries\include"
- Copy "fmodapi375win\api\inc\fmoddyn.h" to "linden\libraries\include"
- Copy "fmodapi375win\api\lib\fmodvc.lib" to "linden\libraries\i686-win32\lib_release" and to "linden\libraries\i686-win32\lib_debug"

(If using cmake, copy "fmodapi375win\api\lib\fmodvc.lib" to "linden\libraries\i686-win32\lib\release" and to "linden\libraries\i686-win32\lib\debug")

- Copy "fmodapi375win\api\fmod.dll" to "linden\indra\newview"

1.5.2. Quicktime

Currently - as of version 1.21 - CMake requires Quicktime to be installed.
Note: This download requires a registration at the Apple Quicktime website and take a bit of time. You can avoid using QuickTime if you want, see this for details. Remember that your Viewer **can't play in-world movies** if you do so.

- Download & install the Quicktime SDK for Windows
- Copy "QuicktimeSDK\Libraries\QTMLClient.lib" to "linden\libraries\i686-win32\lib_release" and to "linden\libraries\i686-win32\lib_debug".

(If using CMake, copy "QuicktimeSDK\Libraries\QTMLClient.lib" to "linden\libraries\i686-win32\lib\release" and to "linden\libraries\i686-win32\lib\debug" instead)

- Copy the contents of "QuicktimeSDK\CIncludes" into "linden\libraries\i686-win32\include\quicktime".

Building with CMake

At this point, you should be ready to use CMake. **NOTE**: CMake is only supported for Viewer versions 1.21 and beyond. Before you first run a build, you'll need to configure things. There's a develop.py script that will create a reasonably sane default configuration for you.
From the command line, **cd into the indra subdirectory** and run one of the following commands (depending on your choice of platform and build environment):

-
  - VisualStudio 2005: "python develop.py -G VC80"
  - VisualStudio 2008: "python develop.py -G VC90"

NOTE: The above commands will configure a "non-standalone" version of the source code tree. This means that the required third party library packages (as built by Linden Lab) will be downloaded during the CMake process.

Finding your build directory

In the CMake world, we keep source and object files separate. The develop.py script will create and populate a build directory for you. It should be in one of the following locations:

- VS 2005: 'build-vc80'
- VS 2008: 'build-vc90'

Compiling

To start a build, do one of the following:

- Run python develop.py build in the indra directory.
- Load it into your IDE. For MSVS VC++:
  - Use "File -> Open -> Project/Solution", and navigate to linden/indra/build-VC80/Viewer.vcproj (or build-VC90, as appropriate).
  - In the MSVS toolbar, just to the right of the triangular "Start Debugging" arrow, is a text box whose tooltip is "Solution Configurations". Select RelWithDebugInfo.
  - If ALL_BUILD is not set as your StartUp Project (the StartUp Project is displayed in bold font), right-click on ALL_BUILD and choose "Set as StartUp Project".
  - Right-click on ALL_BUILD and choose "Properties". In "Configuration Properties -> Debugging", find "Working Directory" and navigate to "linden\indra\newview".
  - Build -> Build Solution (F7)

- Debug -> "Start Debugging" or "Start without debugging".
- MSVC might not be able to find the executable. If not, point it to "linden\indra\build-VC80\newview\relwithdebinfo\secondlife-bin.exe", and try again.
- You may see an error due to not being able to find fmod.dll. If so, find a copy (remember, you copied this in a step above) and copy it into "indra\build-VC80\newview\relwithdebinfo". Try again.
- You may see an error due to not finding llkdu.dll. If so, find it in the normal installed version (make sure it's the same version as your source) and copy it into "indra\build-VC80\newview\relwithdebinfo". Try again.
- Good luck!

Where's the built Viewer?

On Windows, the built Viewer ought to run from VS2005.
To run outside MS VS, see Discussion tab:
Talk:Microsoft_Windows_Builds#Running_Viewer_outside_of_MS_VC

Apéndice B

La información presentada en este apéndice detalla cómo se instala y configura OpenSim.

MS Windows

UPDATED: Feb.15.2010
OpenSim requires either the .Net Framework version 3.51, or Mono 2.4.2.3 or newer. It supports the following compilers:
Net Framework is available from here: www.microsoft.com/downloads/details.aspx

- Microsoft Visual C# Express Edition (note: not Visual C++)
- mono

Additional note: If you like IDE's you will need C# express 2008 or VS 2008.
Additional note: Microsoft C# Express v9 may install .Net 3.5 with resultant path error.

1. To avoid install .Net framework version 2.0 (installed by default in Windows XP & newer)

Additional note: It is possible to develop on Windows Vista 64 bits with the following tweaks:

1. Select OpenSim project properties from solution and choose platform to be x86. Rebuild solution.
2. Select OpenSim.exe properties under solution bin folder and choose windows xp sp 2 compatibility mode + run as administrator.

Building

- In the top-level directory, run the 'runprebuild.bat' file. This will create a VS2008 solution file, a nant build file and a 'compile.bat' file.

- Open the resulting sln file with visual studio and build it there, or
- Run the 'compile.bat' file. This will build the executable using MSBuild.
- if you prefer to use nant, run nant in the same top-level directory. This will build the executables.

If you don't care about physics (walking on prims, etc), ignore the rest of this section.

Running

Recent versions of OpenSim come without an OpenSim.ini file. Copy the OpenSim.ini.example file to OpenSim.ini before making any changes.

Double-click on the OpenSim.exe executable file in the bin directory. This will start up OpenSim in standalone mode.
The debugger in VS2008 C# may be used to step through the code. For those that use a Cygwin shell, you may find that one or more dll's have permissions that cause problems running. Most find that a "chmod 777 *" from the bin directory solves this.
Physics can be invoked by adding the appropriate line to the [Startup] section of OpenSim.ini. For ODE, that would be (DEFAULT in Example):
physics = OpenDynamicsEngine
You can also add a command line option to a shortcut, or run from a command prompt with:
-physics=OpenDynamicsEngine

Windows Vista

Some people have reported that to run on Windows Vista, you must first disable Windows Firewall. Under the new "Start" button of Vista, select "Control panel". Then double-click "Windows Firewall". In the window that pops up, on the left column, select "Turn Windows Firewall on or off". You will have to give permission for this to run, then select the option "Off (not recommended)". Click "OK" and exit from the Windows Firewall window.
If you have McAfee SecurityCenter, see the description below.
Once all the security features are disabled, right click on OpenSim.exe and select "Run as administrator". This will pop up a window asking permission, select "Allow". Your OpenSim server should run in a DOS-like window and accept connections.

McAfee Security

McAfee Security does not allow applications to listen on ports not explicitly specified. You have two options: 1) disable firewall protection all together, 2) enable OpenSim.exe to be able to open ports.

Disable firewall

Open McAfee SecurityCenter. Select "Internet & Network". In the lower left corner is a small link to "Configure...". Select this. In the right side of the window, select the bar that says "Firewall protection is enabled". Here you can select "Off".

Enable OpenSim.exe to open ports

Open McAfee SecurityCenter. Select "Internet & Network". In the lower left corner is a small link to "Configure...". Select this. In the right side of the window, select the bar that says "Firewall protection is enabled". Select the "Advanced..." button. This will pop up a new window.
In the new window, on the left side, select "Program Permissions." In the middle on the right side of the window, select the "Add Allowed Program" button. Use the browser that pops up to find the OpenSim executable and select it.
Finally, select "OK" and exit the McAfee SecurityCenter window.

RECONOCIMIENTOS

Durante el desarrollo de la tesis se tuvo la oportunidad de presentar en las siguientes modalidades (al final se anexan los diplomas):

Artículos:

- Israel Guzmán P., Darnes Vilariño A., Fabiola López L., Maria J. Somodevilla, Mireya Tovar V., Beatriz Beltrán M.(2009). Israel: An agent inside Second Life Virtual World. Congreso Internacional de Ingeniería Electrónica, Biomédica, Computación e Informática. CONCIBE SCIENCE 2009. Publicado en la revista digital www.e-gnosis.udg.mx Page 226. (ver página 60).

- Guzmán I., Vilariño D., López F., Tovar M., Beltrán B. (2008). Diseño de un Agente Inteligente para Mundos Virtuales. Sexto Congreso Nacional de Ciencias de la Computación 2008. CONCIC 2008*.Teoría Algoritmos y Aplicaciones en Computación*. **1**:40-45. (ver página 61).

Ponencias:

- Collaborative and Grid Technologies Workshop 2009, Cancun, Mexico, presented the paper: ISRAEL: AN AGENT INSIDE SECOND LIFE VIRTUAL WORLD Authors: Israel Guzmán P., Darnes Vilariño A., Fabiola López L., Maria J. Somodevilla, Mireya Tovar V., Beatriz Beltrán M. (ver página 62).

- Ponencia en el Simposium de Software Libre, Universidad del Valle de Puebla (UVP), 22 de Septiembre de 2008. Se expuso el primer diseño del agente dentro del mundo virtual y como utilizar herramientas gratuitas para su desarrollo. (ver página 63).

- Ponencia en la Tercera Semana Cultural, Escuela Superior de Ciencias y Humanidades, 27 al 31 de octubre de 2008. Se expuso el diseño del agente dentro del mundo virtual, que son los mundos virtuales y como hacer utilidad de ellos. (ver página 64).

Taller:

- Taller de Second Life (que son mundos virtuales y sus aplicaciones con agentes inteligentes) en la semana de ingenierías “Visión de las ingenierías en el siglo XXI”, Universidad del Valle de Puebla (UVP), 9 de noviembre de 2009. (ver página 65).

La Universidad de Guadalajara,
el Centro Universitario de Ciencias Exactas e Ingenierías,
a través de la División de Electrónica y Computación

otorgan el presente

RECONOCIMIENTO

a

Israel Guzmán P. , Darnes Vilariño A., Fabiola López L., Maria J. Somodevilla, Mireya Tovar V., Beatriz Beltrán M

Por su valiosa participación en la presentación del articulo

ISRAEL: AN AGENT INSIDE SECOND LIFE VIRTUAL WORLD

dentro del marco del

CONGRESO DE COMPUTACION, INFORMATICA, BIOMEDICA Y ELECTRONICA

Dr. Víctor González Álvarez
Rector del CUCEI
Presidente del CONCIBE 2009

CONCIBE 2009

Guadalajara, Jal.,26 al 30 de octubre de 2009

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

Otorga el presente

RECONOCIMIENTO

A

Israel Guzmán Pérez
Darnes Vilariño Ayala
Fabiola López y López
Mireya Tovar Vidal
Beatriz Beltrán Martínez

Por haber presentado la ponencia:

"Diseño de un Agente Inteligente para Mundos Virtuales"

6to Congreso Nacional de Ciencias de la Computación
Del 5 al 7 de noviembre de 2008

SECRETARIA DE INVESTIGACION Y ESTUDIOS DE POSGRADO
FACULTAD CS. DE LA COMP.

Dr. Mario Rossainz López
Director - Fac. de Cs. de la Computación
Universidad Autónoma de Puebla

Dr. Ivo Humberto Pineda Torres
Secretario de Investigación y Estudios de Posgrado

Collaborative and Grid Computing Technologies
Workshop
22nd to 24th April 2009
Cancun, Mexico

RECEIPT AND ATTENDANCE CONFIRMATION LETTER

Registration confirmation 17/4/09

Dear Israel Guzman Perez,

This is confirmation of your attendance at the *Collaborative and Grid Technologies Workshop 2009, Cancun, Mexico,* where you successfully presented the paper: *ISRAEL: AN AGENT INSIDE SECOND LIFE VIRTUAL WORLD* Authors:Israel Guzmán P., Darnes Vilariño A., Fabiola López L., Maria J. Somodevilla, Mireya Tovar V., Beatriz Beltrán M.

We also acknowledge the receipt of payment of the registration fee of £150.00 GBP received on 28 April 2009 from the Benemerita Universidad Autónoma de Puebla, RFC UAP370423PP3, 4 Sur 104 Col. Centro, CP 72000 Puebla, Puebla, México, by The University of Reading on behalf of CGCT 2009.

Thank you,

Linda Mogort-Valls,
CGCT 2009
Collaborative and Grid Technologies Workshop 2009
(CGCT Workshop 2009)
Phone +44 118 378 6372
Fax +44 118 378 5224

Universidad del Valle de Puebla

Otorga el presente

RECONOCIMIENTO

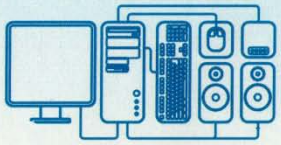
a: **Lic. Israel Guzmán Pérez**

Por su ponencia en el

Simposium de Software Libre

El día 22 de Septiembre de 2008

"Aplicar la ciencia para beneficio del hombre"
H. Puebla de Z. a 22 de Septiembre de 2008



Lic. Concepción Landa Arnaiz
Director del Área de Ingenierías

escihu

Tercera Semana Cultural

somos uno, somos todos, somos ESCIHU

Por su asistencia y participación a la Tercera Semana Cultural que se llevó a cabo del 27 al 31 de octubre del 2008, la Escuela Superior de Ciencias y Humanidades se complace en otorgar a:

Lic. Israel Guzmán Pérez

el siguiente:

reconocimiento

ESTADOS UNIDOS MEXICANOS

SECRETARÍA DE EDUCACIÓN PÚBLICA
SUBSECRETARÍA DE EDUCACIÓN SUPERIOR
DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR
ESCUELA SUPERIOR DE CIENCIAS Y HUMANIDADES
PUEBLA, PUE.
21MSU1160W

Mtra. Martha Lucía Méndez Mena
directora general

ESTADOS UNIDOS MEXICANOS

SECRETARÍA DE EDUCACIÓN PÚBLICA
SUBSECRETARÍA DE EDUCACIÓN SUPERIOR
DIRECCIÓN GENERAL DE
SUPERVISIÓN ESCOLAR 01

Profra. Gloria María Barranco Tapia
supervisora zona 001

escihu cultural tercera semana cultural tercera semana cultural tercera semana cultural tercera semana cultural tercera semana cultura


Universidad del Valle de Puebla
UVP
Universidad del Valle de Puebla
Otorga el presente
Reconocimiento
al: Ing. Israel González Pérez
Por su taller "Second Life"
durante la semana de ingenierías
Visión de las Ingenierías en el siglo XXI
que se llevó a cabo
el 09 de noviembre del presente en esta institución.
"Aplicar la Ciencia para Beneficio del Hombre"
H. Puebla de Z., a 09 de Noviembre de 2009
Lic. Concepción Landa Arnaiz
Directora de la División de Ingenierías