Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

APLICACIÓN DE LAS REDES NEURONALES ARTIFICIALES BASADAS EN EL APRENDIZAJE POR IMITACIÓN EN UN ENTORNO DE ANIMACIÓN VIRTUAL

José Julián Figueroa Leal

Asesorado por el Ing. Herman Igor Véliz Linares

Guatemala, julio de 2013

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

APLICACIÓN DE LAS REDES NEURONALES ARTIFICIALES BASADAS EN EL APRENDIZAJE POR IMITACIÓN EN UN ENTORNO DE ANIMACIÓN VIRTUAL

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JOSÉ JULIÁN FIGUEROA LEAL
ASESORADO POR EL ING. HERMAN IGOR VÉLIZ LINARES

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, JULIO DE 2013

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Walter Rafael Véliz Muñoz
VOCAL V	Br. Sergio Alejandro Donis Soto
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. César Augusto Fernández Cáceres
EXAMINADOR	Ing. Ludwing Federico Altán Sac
EXAMINADOR	Ing. César Rolando Batz Saquimux
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

APLICACIÓN DE LAS REDES NEURONALES ARTIFICIALES BASADAS EN EL APRENDIZAJE POR IMITACIÓN EN UN ENTORNO DE ANIMACIÓN VIRTUAL

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha noviembre de 2011.

José Julian Figueroa Leal

Guatemala 15 de Diciembre de 2011

Ingeniero
Marlon Pérez Turk
Director de escuela
Escuela de Ingeniería en Ciencias y Sistemas
Facultad de ingeniería, USAC

Señor director:

Atentamente me dirijo a usted para informarle que he tenido a bien asesorar el trabajo de tesis: **"APLICACIÓN DE LAS REDES NEURONALES ARTIFICIALES BASADAS EN EL APRENDIZAJE POR IMITACIÓN EN UN ENTORNO DE ANIMACIÓN VIRTUAL",** realizado por el estudiante **José Julián Figueroa Leal**, quien se identifica con carné **No. 2003-12474**; previo a optar el título de Ingeniero en ciencias y sistemas.

Al respecto quiero indicarle que luego de efectuadas las revisiones y correcciones del caso, encuentro satisfactorio el trabajo por lo que procedo a aprobarlo y remitirlo a usted para su trámite correspondiente.

Atentamente,

Ing. Herman Igor Véliz Linares
COLEGIADO No. 4836

Ing. Herman Igor Véliz Linares
Asesor

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 08 de Febrero de 2012

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **JOSÉ JULIÁN FIGUEROA LEAL** carné **2003-12474,** titulado: **"APLICACIÓN DE LAS REDES NEURONALES ARTIFICIALES BASADAS EN EL APRENDIZAJE POR IMITACIÓN EN UN ENTORNO DE ANIMACIÓN VIRTUAL",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
INGENIERIA EN CIENCIAS Y SISTEMAS
COORDINACION

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"APLICACIÓN DE LAS REDES NEURONALES ARTIFICIALES BASADAS EN EL APRENDIZAJE POR IMITACIÓN EN UN ENTORNO DE ANIMACIÓN VIRTUAL"****, realizado por el estudiante JOSÉ JULIÁN FIGUEROA LEAL, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Turk
Director, Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 16 de julio 2013

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

DTG. 508.2013

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **APLICACIÓN DE LAS REDES NEURONALES ARTIFICIALES BASADAS EN EL APRENDIZAJE POR IMITACIÓN EN UN ENTORNO DE ANIMACIÓN VIRTUAL,** presentado por el estudiante universitario: **José Julián Figueroa Leal,** autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Murphy Olympo Paiz Recinos
Decano

Guatemala, 18 de julio de 2013

/gdech

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

ACTO QUE DEDICO A:

Dios	Por proveerme esperanza y fuerza en la realización de este logro.
Mis padres	Alfredo Figueroa y Lilian de Figueroa, por ser ejemplo de honestidad, integridad, esfuerzo, benignidad, apoyo, lucha, amor inagotable y fuente de mi orgullo.
Mi hermana	Mariela Figueroa, compañera admirable de mis proyectos y experiencias personales, en las buenas y en las malas.

AGRADECIMIENTOS A:

Dios	Quien me proporcionó las fuerzas e inteligencia para la realización de este trabajo.
Mis padres	Alfredo Figueroa y Lilian de Figueroa, por su maravilloso e interminable amor y apoyo que para mí es el verdadero corazón y núcleo de este logro.
Mi hermana	Mariela Figueroa, por su incuestionable ímpetu en apoyarme para alcanzar esta meta.
Mi asesor	Ing. Herman Véliz, por su tiempo invertido en asesorar este trabajo y por su apoyo para la realización del mismo.
Universidad de San Carlos de Guatemala	El más competente centro de estudios de Guatemala.
Todas las personas	Que ayudaron en la realización de este trabajo.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
&	Indica que se accede a una posición de memoria en el lenguaje c++
∑	Sumatoria de todos los elementos

GLOSARIO

Axón	Salida de una neurona que conecta con otras dentro de la red neuronal.
Dentritas	Extensiones de las neuronas utilizadas para establecer conexiones con otras neuronas.
Experimento del muñeco Bobo	Experimento realizado por Albert Bandura que demostraba el inherente proceso de aprendizaje por imitación en el ser humano.
Frame	Se denomina frame en inglés, a un fotograma o cuadro, una imagen particular dentro de una sucesión de imágenes que componen una animación. La continua sucesión de estos fotogramas producen a la vista la sensación de movimiento, fenómeno dado por las pequeñas diferencias que hay entre cada uno de ellos

Función de activación	Criterio de evaluación de la salida que se genera en la neurona en base a las entradas de esta.
Havok	Havok (Havok Physics) es un motor físico middleware desarrollado por una compañía irlandesa con el mismo nombre, que por medio de simulación dinámica permite recrear interacciones de carácter físico en videojuegos y otras aplicaciones multimedia. Actualmente sus librerías se encuentran publicadas para propósitos no comerciales.
Matriz de posición	Matriz de coordenadas cartesianas que representan las posiciones de los vértices, cada uno respecto de su vértice de referencia.
Matriz de posición resultante	Matriz de posición donde se identifico un cambio en el flujo del movimiento del modelo humanoide.
Matriz de secuencia	Almacena un conjunto de matrices resultantes con la información del tiempo en el que se identificaron en términos de *frames*.

Modelo Perceptrón multicapa	Modelo de red neuronal artificial compuesto por varias capas organizadas en niveles que se interconectan de manera total o de manera parcial
Modelo Rosenblatt	Modelo de red neuronal creado inicialmente para ilustrar propiedades fundamentales de sistemas inteligentes en general, inventado por el psicólogo Frank Rosenblatt en el 1957.
Neurona	Las neuronas son un tipo de células del sistema nervioso cuya principal característica el procesamiento de información dentro del cerebro por medio de impulsos nerviosos entre ellas, formando así redes neuronales; en este trabajo se emplea el uso de neuronas artificiales que son representaciones de software de las neuronas biológicas.
Red neuronal especular artificial	Red neuronal que aprende el comportamiento de unmodelo tridimensional en base a un modelo perceptrón multicapa.

Zona de asociación	Zona de procesado de la información obtenida en la zona sensorial en el modelo Rosenblatt.
Zona de respuesta	Zona de interpretación de los resultados obtenidos en la zona de asociación.
Zona sensorial	Zona del modelo Rosenblatt dedicada a la captura de la información a procesar.

RESUMEN

Albert Bandura (Canadá, 4 de diciembre de 1925) psicólogo ucraniano-canadiense, famoso por su trabajo respecto de la teoría de aprendizaje social. Es conocido particularmente por el experimento del muñeco Bobo sobre el comportamiento agresivo de los niños. De la inmensa cantidad de estudios de Bandura, se destaca un grupo de ellos encima de los demás: los estudios del muñeco bobo (un muñeco bobo es un juguete que posee forma de huevo con cierto peso en su base que hace que se tambalee cuando se le golpea. Estos estudios, Bandura los realizó partiendo de una película, donde una joven estudiante solo pegaba a un muñeco bobo.

La joven golpeaba al muñeco, y al mismo tiempo gritando ¡estúpidooooo!. Se sentaba encima de él, lo golpeaba con un martillo, además de diversas acciones violentas y gritando variedad de frases agresivas. Bandura mostró la película a un grupo de niños de guardería. Posteriormente se les dejó jugar. En el salón de juegos, por supuesto, había varios observadores con bolígrafos y carpetas, un muñeco bobo nuevo y algunos pequeños martillos. Lo que los observadores anotaron fue: un gran coro de niños golpeando a descaro al muñeco bobo. Le pegaban gritando ¡estúpidooooo!, se sentaron sobre él, lo golpearon con los martillos, etc. En otras palabras, imitaron a la joven de la película y de una manera bastante precisa. A simple vista, parece un experimento con poco de aportación en principio, pero considérese por un momento que estos niños cambiaron su comportamiento ¡sin que hubiese inicialmente un refuerzo dirigido a explotar dicho comportamiento! Y aunque esto no parezca extraordinario para cualquier padre,

maestro o un observador casual de niños, no encajaba muy bien con las teorías de aprendizaje conductuales estándares.

Tomando en cuenta que estos procesos de aprendizaje del niño, son procesos químicos y biológicos, no son procesos electrónicos, sino siendo el cerebro humano un procesador de información de gran inteligencia con un funcionamiento químico biológico.

Bandura llamó a este fenómeno, aprendizaje por observación o modelado, y su teoría usualmente se conoce como la Teoría Social del Aprendizaje.

Aún más interesante que el experimento de Bandura, sería repetirlo usando inteligencia artificial, si existiera un software que puesto en un robot humanoide, que le permitiera imitar el comportamiento de los seres humanos, la cantidad de aprendizaje del software sería increíblemente significativo. Sin embargo actualmente realizar investigaciones teniendo a la mano semejante hardware ya es bastante complicado, y los sensores actuales aún no son tan desarrollados como los sentidos humanos para proveer toda la información que estos proporcionan y en la forma que se proporciona.

Sin embargo, en un entorno virtual basado en animación es posible realizar investigaciones respecto de cómo el software puede imitar un comportamiento, porque en este entorno es perfectamente manipulable y se pueden programar los sensores dentro de él, sin invertir poco más que el esfuerzo necesario para su desarrollo. Esta tesis proporciona el análisis y diseño de un software que simula el proceso de aprendizaje por imitación reconocido por Bandura en un entorno basado en animación tridimensional, explicado a lo largo de todo el documento.

OBJETIVOS

General

Analizar profundamente la capacidad de aprendizaje por imitación que el software puede lograr utilizando una estructura de redes neuronales artificiales, usando la tecnología actual en un entorno virtual que tenga como plataforma la animación 3D como herramienta para la fácil abstracción de la solución del problema.

Específicos

1. Estudiar el proceso de aprendizaje natural por imitación del ser humano como base para el análisis y diseño del software, y la capacidad de aprendizaje que puede lograr.

2. Establecer el análisis y diseño del software que logre realizar aprendizaje por imitación en un entorno virtual basado en animación 3D.

3. Conocer las ventajas que proporciona un entorno basado en animación 3D para la investigación de la inteligencia artificial.

4. Demostrar que es posible proveer de aprendizaje por imitación a un modelo humanoide utilizando una red neuronal artificial.

5. Dar a conocer las tecnologías actuales en desarrollo de videojuegos aplicadas a la inteligencia artificial.

6. Establecer las bases para el desarrollo de software que represente los procesos de aprendizaje por imitación del ser humano.

7. Dar a conocer la fuerte relación que posee la investigación en inteligencia artificial con la psicología y los procesos de aprendizaje natural.

INTRODUCCIÓN

Se define el aprendizaje como un proceso que consiste en cambios de adaptación por parte de la conducta individual resultando de la experiencia obtenido en cierto entorno. El propósito de este documento es estudiar como el software puede lograr la capacidad de aprendizaje por imitación mediante redes neuronales artificiales dentro de un entorno tridimensional.

Basándose la observación y el análisis, múltiples psicólogos y pedagogos han llegado a definir el aprendizaje así como identificar distintos tipos de aprendizaje, por ejemplo, es común encontrarse en una situación en donde un niño suele imitar las acciones de los adultos o repetir las palabras o frases de otra persona, este proceso de aprendizaje por imitación es enormemente intensivo en la etapa de la infancia, y muy importante para el desarrollo del cerebro humano, y sigue latente a lo largo de la vida humana, conocer qué tipo de aprendizaje represente y las teorías actuales de cómo se realiza este proceso de aprendizaje, y cómo el software puede simularlo en un entorno tridimensional es el enfoque directo de el tema de tesis para el cual se realiza este documento.

1. APRENDIZAJE NATURAL

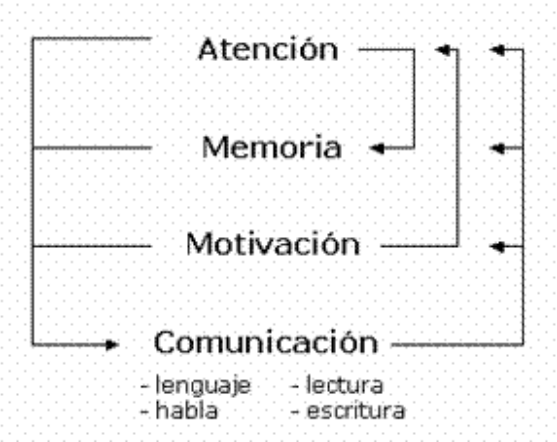
La capacidad del cerebro humano para procesar enormes cantidades de información usando solamente procesos químicos y biológicos es en gran manera sorprendente, aunque los computadores pueden realizar al día de hoy de una manera mucho más eficiente que el cerebro humano diversas tareas, todavía no tienen la capacidad de manejar toda la información que llega de los cinco sentidos humanos simultáneamente en la manera que lo hace el cerebro humano, y la manera en que se coordinan todas las acciones y reacciones del cuerpo en base a esta información, esto también considerando que los cinco sentidos funcionan como sensores del cerebro humano de una increíble capacidad de precisión en cuanto a lo que en la realidad está ocurriendo. Tomando en cuenta todo esto, los computadores y el software actual ya no se muestran tan abrumadoramente capaces como pueden llegar a parecer, esto es más notable al momento de realizar tareas relacionadas al aprendizaje relacionado con situaciones dinámicas y diferentes en entornos diferentes y no especificas.

1.1. ¿Pero qué es el aprendizaje?

El aprendizaje puede definirse como un proceso del cerebro que captura información, la almacena, la relaciona, la asocia, para poder utilizarla cuando sea necesaria, el uso de esta información puede ser mental como por ejemplo un recuerdo, concepto o dato, pero este uso también puede ser instrumental como la realización de una tarea. En cualquier caso es por medio de los sentidos que esta información es aprendida.

Existen cuatro procesos que se consideran esenciales, tal como aparecen en la siguiente figura, son la atención, la memoria, la motivación y la comunicación

Figura 1. **Las claves del aprendizaje**



Fuente: http://www.down21.org/salud/neurobiologia/bases_aprend.htm. Consulta: 15 de diciembre de 2011.

En este esquema se puede constatar la estrecha interrelación que existe entre los cuatro procesos: son tanto más esenciales e importantes cuanto más se asciende en la escala de las especies, de modo que alcanzan su máxima expresión e importancia en el ser humano.

1.1.1. Atención

El grado en que se le pone atención a algo es muy importante para determinar que tanto se absorbe la información para poder aprenderla. En la atención están involucrados varias áreas y núcleos del cerebro. Algunas áreas están relacionadas con la responsabilidad de recibir y, sobre todo, de integrar la información que llegan a través de los sentidos. Otras están relacionados

con la retención inmediata de la información para saber su significado, y para contrastar su importancia, como por ejemplo se puede manejar la información respecto a ciertos indicadores como lo que podrían representarse mediante las siguientes preguntas: ¿es nueva o ya conocida?, ¿vale la pena retenerla?, ¿vale la pena seguir recibiéndola?, ¿me interesa?. Otras están encargadas de rechazar y filtrar todo aquello que pueda distraer y cambiar el objeto de atención actual.

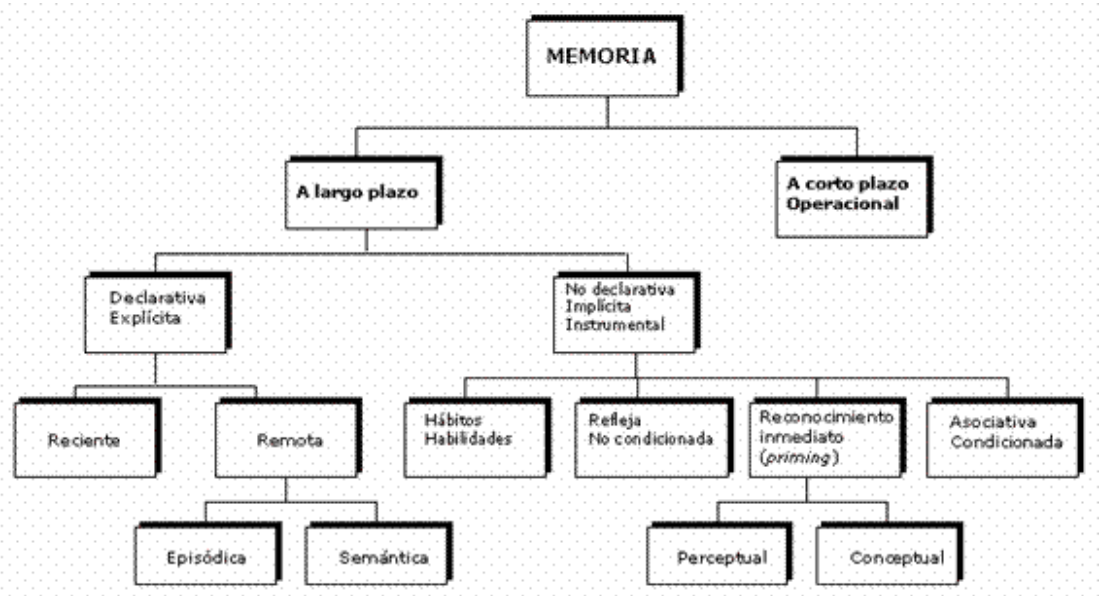
1.1.2. Memoria

La memoria es un proceso del cerebro que permite codificar, registrar, consolidar y almacenar la información de manera que, cuando se necesite, pueda accederse a ella y evocarla. Así es en gran manera esencial para el aprendizaje. La memoria adopta distintas formas que son definidas por distintas estructuras cerebrales. Pueden mencionarse dos grandes tipos.

- La que es llamada de corto plazo o de corta duración, inmediata, operacional.
- La que es llamada de largo plazo o de larga duración que, a su vez, es dividida en otras dos:
  - La declarativa o explícita, que puede ser episódica o semántica
  - La no declarativa, implícita, instrumental o procedimental.

A continuación se muestra una imagen es un diagrama de la organización de las distintas formas que la memoria puede adoptar.

Figura 2. **Organización de memoria**



Fuente: http://www.down21.org/salud/neurobiologia/bases_aprend.htm. Consulta: 15 de diciembre de 2011.

Uno de los propósitos de esta investigación es hacer especial énfasis en el aprendizaje relacionado con la conducta y con las habilidades motoras en asociación con un entorno, es por esto que a continuación se define únicamente el tipo de memoria instrumental.

La memoria instrumental o de procedimiento tiene relación con la capacidad para aprender habilidades expresadas en forma de conducta o comportamiento, cognitivas y normativas, que se utilizan para realizar actividades de manera automática o hasta inconsciente. De los núcleos cerebrales con la responsabilidad de esta memoria son las áreas motoras. Cuando esta memoria se pierde, la persona empieza por olvidar habilidades

elementales de aseo persona, escribir, tocar un instrumento, conducir un coche, prepararse un plato.

1.1.3. Motivación

Es la propiedad que impulsa y capacita para ejecutar una actividad. Por eso se encuentra tanto en la base de atención (porque si no se está motivado no se mantendrá la atención y menos aún se llegará a enfrascarse), como en la base de la memoria (como elemento de reforzamiento importantísimo: cómo se recuerda lo que más afecta), y en la base de la realización de cualquier actividad: se impulsa a la acción. Más adelante se podrá ver como la motivación puede formar parte en el diseño y análisis del software ejemplo.

1.1.4. Comunicación

La comunicación es esencial para captar cualquier tipo de información, sea visual o auditiva o incluso verbal, y por consiguiente, para aprenderla. Pero en la especie humana, la comunicación en cualquiera de sus formas ha adquirido tal grado de protagonismo que se ha convertido en elemento que influye de modo decisivo sobre los otros tres grandes procesos del aprendizaje. Por eso, la comunicación necesita de amplias zonas del cerebro y de complicados mecanismos de funcionamiento que aseguren la comprensión y la expresión de lo comunicado, sea a través de la expresión corporal y gestual, o del lenguaje en sus variadas formas, de las que el oral es muy importante pero no el único. Por esto al diseñar un software que aprenda es de gran importancia la forma en que este se va a comunicar con la realidad con la que va a coexistir.

En conclusión, sentadas las bases del aprendizaje (atención, memoria, motivación y comunicación), las condiciones ahora permiten enfocarse más en el aprendizaje por imitación presente en el ser humano, como fundamento para el análisis y diseño de un software que pueda aprender de una forma muy similar

1.2. Teoría de la Personalidad

Albert Bandura es un reconocido psicólogo graduado de la Universidad de Columbia Británica en 1949, que además de recibir el premio por las Contribuciones Científicas Distinguidas en 1980, se le atribuye ser uno de los máximos exponentes de las teorías de la personalidad. A continuación se describe de manera breve su teoría concerniente al aprendizaje por observación o teoría de la personalidad

1.2.1. Teoría de la Personalidad de Albert Bandura

El conductismo, se focaliza sobre variables que pueden observarse, medirse y manipularse y rechaza todo aquello que sea subjetivo, interno y no disponible. En el método experimental, el procedimiento estándar es manipular una variable y luego medir sus efectos sobre otra. Todo esto conlleva a una teoría de la personalidad que dice que el entorno del individuo causa su comportamiento.

Bandura consideró que esto era un poquito simple para el fenómeno que observaba (agresión en adolescentes) y por tanto decidió añadir un poco más a la fórmula: sugirió que el ambiente causa el comportamiento; cierto, pero que el comportamiento causa el ambiente también. Definió este concepto

con el nombre de determinismo recíproco: el mundo y el comportamiento de una persona se acusan mutuamente

Más tarde, fue un paso más allá. Empezó a considerar a la personalidad como una interacción entre tres cosas: el ambiente, el comportamiento y los procesos psicológicos de la persona. Estos procesos consisten en la habilidad para abrigar imágenes en la mente y en el lenguaje. Desde el momento en que introduce la imaginación en particular, deja de ser un conductista estricto y empieza a acercarse a los cognitivistas. De hecho, usualmente es considerado el padre del movimiento cognitivo.

1.2.2. Aprendizaje por la observación o modelado

De los cientos de estudios de Bandura, un grupo se alza por encima de los demás, los estudios del muñeco bobo. Lo hizo a partir de una película de uno de sus estudiantes, donde una joven estudiante solo pegaba a un muñeco bobo. En caso de que no se sepa, un muñeco bobo es una criatura hinchable en forma de huevo con cierto peso en su base que hace que se tambalee cuando se le pega. En aquella época llevaba pintado al payaso Bobo de protagonista.

La joven pegaba al muñeco, gritando ¡estúpidooooo!. Le pegaba, se sentaba encima de él, le daba con un martillo y demás acciones gritando varias frases agresivas. Bandura les enseñó la película a un grupo de niños de guardería que, como podrá suponerse, saltaron de alegría al verla. Posteriormente se les dejó jugar. En el salón de juegos, por supuesto, había varios observadores con bolígrafos y carpetas, un muñeco bobo nuevo y algunos pequeños martillos.

Aquí es posible predecir lo que los observadores anotaron: un gran coro de niños golpeando a descaro al muñeco bobo. Le pegaban gritando ¡ estúpidooooo!, se sentaron sobre él, le pegaron con martillos y demás. En otras palabras, imitaron a la joven de la película y de una manera bastante precisa.

Esto podría parecer un experimento con poco de aportación en principio, pero debe considerarse un momento: estos niños cambiaron su comportamiento ¡sin que hubiese inicialmente un refuerzo dirigido a explotar dicho comportamiento! Y aunque esto no parezca extraordinario para cualquier padre, maestro o un observador casual de niños, no encajaba muy bien con las teorías de aprendizaje conductuales estándares. Bandura llamó al fenómeno aprendizaje por la observación o modelado, y su teoría usualmente se conoce como la teoría social del aprendizaje.

Bandura llevó a cabo un largo número de variaciones sobre el estudio en cuestión: el modelo era recompensado o castigado de diversas formas de diferentes maneras; los niños eran recompensados por sus imitaciones; el modelo se cambiaba por otro menos atractivo o menos prestigioso y así sucesivamente. En respuesta a la crítica de que el muñeco bobo estaba hecho para ser pegado, Bandura incluso rodó una película donde una chica pegaba a un payaso de verdad. Cuando los niños fueron conducidos al otro cuarto de juegos, encontraron lo que andaban buscando… ¡un payaso real! Procedieron a darle patadas, golpearle, darle con un martillo, etc.

Todas estas variantes permitieron a Bandura a establecer que existen ciertos pasos envueltos en el proceso de modelado:

1.2.2.1. Atención

Si se va a aprender algo, se necesita estar prestando atención. De la misma manera, todo aquello que suponga un freno a la atención, resultará en un detrimento del aprendizaje, incluyendo el aprendizaje por observación. Si por ejemplo, se está adormilado, drogado, enfermo, nervioso, etc., se aprenderá menos. Igualmente ocurre si se está distraído por un estímulo competitivo.

Alguna de las cosas que influye sobre la atención tiene que ver con las propiedades del modelo. Si el modelo es colorido y dramático, por ejemplo, se presta más atención. Si el modelo es atractivo o prestigioso o parece ser particularmente competente, se presta más atención. Y si el modelo se parece más a un ser humano, ser prestara más atención. Este tipo de variables encaminó a Bandura hacia el examen de la televisión y sus efectos sobre los niños.

1.2.2.2. Retención

Segundo, se debe ser capaz de retener (recordar) aquello a lo que le se ha prestado atención. Aquí es donde la imaginación y el lenguaje entran en juego: se guarda lo que se ha visto hacer al modelo en forma de imágenes mentales o descripciones verbales. Una vez archivados, puede hacerse resurgir la imagen o descripción de manera que pueda reproducirse con el propio comportamiento.

1.2.2.3. Reproducción

En este punto, se está ahí soñando despierto. Se deben traducir las imágenes o descripciones al comportamiento actual. Por tanto, lo primero de lo que se debe ser capaz es de reproducir el comportamiento. Se puede pasar todo un día viendo a un patinador olímpico haciendo su trabajo y no poder ser capaz de reproducir sus saltos, ya que ¡no sé sabe nada de patinar! Por otra parte, si se pudiera patinar, la demostración de hecho mejoraría si se observa a patinadores mucho más hábiles.

Otra cuestión importante con respecto a la reproducción es que la habilidad para imitar mejora con la práctica de los comportamientos envueltos en la tarea. Y otra cosa más: las habilidades mejoran ¡aún con el solo hecho de imaginar que se está haciendo el comportamiento! Muchos atletas, por ejemplo, se imaginan el acto que van a hacer antes de llevarlo a cabo.

1.2.2.4. Motivación

Aún con todo esto, todavía no se hace nada a menos que se esté motivado a imitar; es decir, a menos que se tengan buenas razones para hacerlo. Bandura menciona un número de motivos:

- Refuerzo pasado, como el conductismo tradicional o clásico.
- Refuerzos prometidos, (incentivos) que se puedan imaginar.
- Refuerzo vicario, la posibilidad de percibir y recuperar el modelo como reforzador.

Nótese que estos motivos han sido tradicionalmente considerados como aquellas cosas que causan el aprendizaje. Bandura especifica que éstos no son tan causantes como muestras de lo que se ha aprendido. Es decir, él los considera más como motivos. Por supuesto que las motivaciones negativas también existen, dando motivos para no imitar:

- Castigo pasado
- Castigo prometido (amenazas)
- Castigo vicario

Como la mayoría de los conductistas clásicos, Bandura dice que el castigo en sus diferentes formas no funciona tan bien como el refuerzo y, de hecho, tiene la tendencia a volverse contra el individuo.

1.2.3. Autorregulación

La autorregulación (controlar el propio comportamiento) es la otra piedra angular de la personalidad humana. En este caso, Bandura sugiere tres pasos explicados desde una primera persona para comprenderlos más fuertemente:

A. Auto-observación. El individuo se ve a sí mismo, su comportamiento y recolecta pistas de ello.

B. Juicio. El individuo compara lo que ve con un estándar. Por ejemplo, puede comparar sus actos con otros tradicionalmente establecidos, tales como reglas de etiqueta. O puede crear algunos nuevos, como leeré un libro a la semana. O puede competir con otros, o consigo mismo.

C. Auto-respuesta. Si se ha salido bien en la comparación con el estándar, el individuo se da respuestas de recompensa a sí mismo. De lo contrario, se dará autorespuestas de castigo. Estas auto-respuestas pueden ir desde el extremo más obvio (decirse algo malo o trabajar hasta tarde), hasta el otro más encubierto (sentimientos de orgullo o vergüenza).

Un concepto muy importante en psicología que podría entenderse bien con la autorregulación es el autoconcepto (mejor conocido como autoestima). Si a través de los años, se ve que se ha actuado más o menos de acuerdo con ciertos estándares y se ha tenido una vida llena de recompensas y alabanzas personales, se tendrá un auto-concepto agradable (autoestima alta). Si, de lo contrario, se ha visto incapacidad de alcanzar los estándares y castigándose por ello, se tendrá un pobre autoconcepto (autoestima baja)

1.2.4. La terapia de modelado

La terapia por la que Bandura es más conocido es la del modelado. Esta teoría sugiere que si se escoge a alguien con algún trastorno psicológico y se le hace observar a otro que está intentando lidiar con problemas similares de manera más productiva, el primero aprenderá por imitación del segundo.

La investigación original de Bandura sobre el particular envuelve el trabajo con herpefóbicos (personas con miedos neuróticos a las serpientes) El cliente es conducido a observar a través de un cristal que da a un laboratorio.

En este espacio, no hay nada más que una silla, una mesa, una caja encima de la mesa con un candado y una serpiente claramente visible en su interior. Luego, la persona en cuestión ve cómo se acerca otra (un actor) que se dirige lenta y temerosamente hacia la caja. Al principio actúa de forma muy aterradora; se sacude varias veces, se dice a sí mismo que se relaje y que respire con tranquilidad y da un paso a la vez hacia la serpiente. Puede detenerse en el camino un par de veces; retraerse en pánico, y vuelve a empezar. Al final, llega al punto de abrir la caja, coge a la serpiente, se sienta en la silla y la agarra por el cuello; todo esto al tiempo que se relaja y se da instrucciones de calma.

Después que el cliente ha visto todo esto (sin duda, con su boca abierta durante toda la observación), se le invita a que él mismo lo intente. Considerando que, él sabe que la otra persona es un actor (¡no hay decepción aquí; solo modelado!) Y aún así, muchas personas, fóbicos crónicos, se embarcan en la rutina completa desde el primer intento, incluso cuando han visto la escena solo una vez. Esta desde luego, es una terapia poderosa.

Una pega de la terapia era que no es tan fácil conseguir las habitaciones, las serpientes, los actores, etc., todos juntos. De manera que Bandura y sus estudiantes probaron diferentes versiones de la terapia utilizando grabaciones de actores e incluso apelaron a la imaginación de la escena bajo la tutela de terapeutas. Estos métodos funcionaron casi tan bien como el original.

Albert Bandura tuvo un enorme impacto en las teorías de la personalidad y en la terapia. Su estilo lanzado y parecido al de los conductistas les pareció bastante lógico a la mayoría de las personas. Su acercamiento

orientado a la acción y a la solución de problemas era bienvenido por aquellos que les gustaba la acción más que filosofar sobre el ello, arquetipos, actualización, libertad y todos los otros constructos mentalistas que los personólogos tienden a estudiar.

Dentro de los psicólogos académicos, la investigación es crucial y el conductismo ha sido su acercamiento preferido. Desde los últimos años de los 60, el conductismo ha dado paso a la revolución cognitiva, de la cual Bandura es considerado parte. La psicología cognitiva retiene el sabor de la orientación experimental del conductismo, sin retener artificialmente al investigador de comportamientos externos, cuando precisamente la vida mental de los clientes y sujetos es tan obviamente importante.

Este es un movimiento poderoso, y sus contribuyentes incluyen a algunas de las personas más destacadas en la psicología actual, como por ejemplo: Julián Rotter, Walter Mischel, Michael Mahoney y David Meichenbaum. También hay otros dedicados a la terapia como Beck (terapia cognitiva) y Ellis (terapia racional- emotiva) Los seguidores y posteriores a George Kelly también se encuentran en este campo. Y las muchas otras personas que se están ocupando del estudio de la personalidad desde el punto de vista de los rasgos, como Buss y Plomin (teoría del temperamento) y McCrae y Costa (teoría de los cinco factores) son esencialmente conductistas cognitivos como Bandura.

1.3. Del aprendizaje natural al aprendizaje artificial

El desarrollo de un modelo de aprendizaje natural incluye simular los logros del aprendizaje natural, pero a su vez implica también simular sus mismos errores, es decir, simular el proceso de aprendizaje en sí mismo. Es por

esto que a menudo se tiende a pensar que lo que para un sistema de aprendizaje natural puede ser importante, no tiene por qué serlo para un sistema artificial, o viceversa, sin embargo muchos de esos errores son asumidos por los sistemas naturales como un pequeño precio a pagar a cambio de mantener un sistema de aprendizaje flexible y rápido que permita sobrevivir y adaptarse de manera eficaz a los cambios ambientales. Es más, como se mostrará a continuación, ese comportamiento tan poco lógico que a menudo muestran los sistemas naturales no es en realidad un error de la naturaleza, sino más bien una forma sabia de adaptarse al ambiente.

Los modelos teóricos de aprendizaje natural desarrollados en la psicología del aprendizaje son parecidos a los sistemas de aprendizaje desarrollados en inteligencia artificial. Pero a diferencia de los sistemas artificiales, los modelos de aprendizaje natural tienen como objetivo el lograr simular y predecir el aprendizaje que realizan los seres humanos y los demás animales teniendo en cuenta no sólo sus limitaciones de memoria y de capacidad de procesamiento, sino también el tiempo limitado del que disponen a la hora de responder a las demandas de su ambiente. Del diálogo entre los investigadores del aprendizaje natural y el artificial podrían surgir sistemas artificiales que sean capaces de adaptarse eficazmente a los cambios de su entorno.

Puede verse el siguiente ejemplo concreto: Tomese un perro de Pavlov al que se le presenta un sonido seguido por comida durante veinte ensayos. El animal aprende que el sonido predice la comida, y como consecuencia de ello, acaba salivando nada más escuchar el sonido. La salivación es lo que indica que ha aprendido esa relación predictiva. Pero si ahora se le presenta el sonido sólo (sin comida) otras veinte veces, el animal dejará de salivar. ¿Qué significa eso, que ha desaprendido lo que ya sabía? No.

Sencillamente ha aprendido una relación opuesta a la anterior y sabe que ahora, el sonido ya no predice comida.

Este segundo aprendizaje interfiere con el primero, por eso el perro ya no saliva cuando escucha el sonido. Y debe ser así, esto es lo único que tiene sentido en la naturaleza. Y sin embargo, esta interferencia del nuevo aprendizaje sobre el más antiguo podría ser considerado como un error fatal si sólo se toman en cuanta las reglas de la lógica, según las cuales el animal debería al final concluir que la comida seguirá al sonido en el
50% de los ensayos.

¿Por qué no saliva, entonces? Porque en el contexto actual, lo que tiene sentido es no esperar la comida. Pero lo más interesante es que esa interferencia que ocurre en el aprendizaje natural nunca es lo que se conoce en muchos sistemas artificiales como interferencia catastrófica: Bastará con dejar pasar un cierto periodo de tiempo para que el animal deje de sentirse en el contexto en el que recibió los últimos ensayos y, por tanto, vuelva a salivar al escuchar el sonido. Esto demuestra que no había olvidado lo aprendido, y de hecho, lo recupera fácilmente, pero sólo cuando tiene sentido hacerlo. Lo mismo ocurrirá, por supuesto, si lo que se manipula son contextos físicos en vez de temporales. Es decir, el perro no sólo es capaz de aprender la información contradictoria sin destruir el conocimiento previo sino que utiliza uno u otro tipo de información de manera flexible según el contexto en el que se encuentre (el sonido significa comida en el contexto A pero no en el B y el perro lo sabe, por eso salivará sólo en el contexto A). Las personas funcionan de manera similar cuando aprenden, por ejemplo, a diferenciar lo que significa una palabra en un contexto o en otro, pero el modelo animal proporciona una situación simplificada en la que a veces resulta más sencillo el desarrollo teórico y la simulación del proceso.

Lo anterior es un ejemplo de los muchos datos y teorías provenientes del aprendizaje natural que podrían ser aplicados al aprendizaje artificial. Es relativamente sencillo simular este tipo de aprendizaje y una máquina con capacidad de aprendizaje tendría que ser capaz tanto de mostrar el efecto de interferencia del nuevo conocimiento sobre el más antiguo, como de recuperar el conocimiento más antiguo cuando las demandas del ambiente sugieran que ese tipo de estrategia será más ventajosa.

Los humanos y los demás animales aprenden de manera muy flexible, adaptándose a cambios que tienen lugar en el ambiente, y es precisamente esta capacidad de cambiar y de adaptarse al ambiente lo que les permite sobrevivir. Esto, además, debe hacerse constantemente y de manera muy rápida, normalmente sin demasiado tiempo para pensar o realizar cálculos complicados, incluso sin tener en cuenta todas las opciones posibles a medida que se va tomando decisiones con la información que se reciben de los sentidos. Pero precisamente la capacidad de adaptación radica ahí, en la ausencia de comportamientos rígidos, en la capacidad de asumir riesgos y de dar respuestas más o menos correctas a partir de ciertas informaciones. Este diseño es muy arriesgado pero completamente funcional.

Es cierto que aún no es posible construir una máquina que sea capaz de aprender igual que como lo hacen los seres humanos y los demás animales... sencillamente porque no se tiene aún una teoría lo suficientemente buena como para que pueda ser aplicada con éxito a todas las áreas del aprendizaje natural. Pero sí es posible aplicar las actuales teorías psicológicas del aprendizaje a un número cada vez mayor de situaciones concretas. Del diálogo entre los investigadores del aprendizaje natural y el aprendizaje

artificial pueden surgir tanto modelos teóricos más perfectos del aprendizaje natural como sistemas artificiales que sean capaces de adaptarse eficazmente a los cambios de su entorno

2. REDES NEURONALES ARTIFICIALES

Las Redes Neuronales Artificiales (ANNs de Artificial Neural Networks) fueron inicialmente una simulación abstracta de los sistemas nerviosos biológicos, formados conjuntos de unidades llamadas neuronas o nodos interconectadas entre sí. Estas conexiones tienen una gran semejanza con las dendritas y los axones en los sistemas nerviosos biológicos.

Se puede considerar una primera clasificación de los modelos de redes neuronales artificiales, atendiendo a su similitud con la realidad biológica los siguientes dos tipos:

A. Los modelos de tipo biológico. Comprende las redes que tratan de simular los sistemas neuronales biológicos así como las funciones auditivas o algunas funciones básicas de la visión.

B. El modelo dirigido a aplicación. Sus arquitecturas están fuertemente relacionadas a las necesidades de las aplicaciones para las que son diseñados. Estos modelos no tienen porque guardar algún parecido o similitud con los sistemas biológicos

2.1. Redes neuronales de tipo biológico

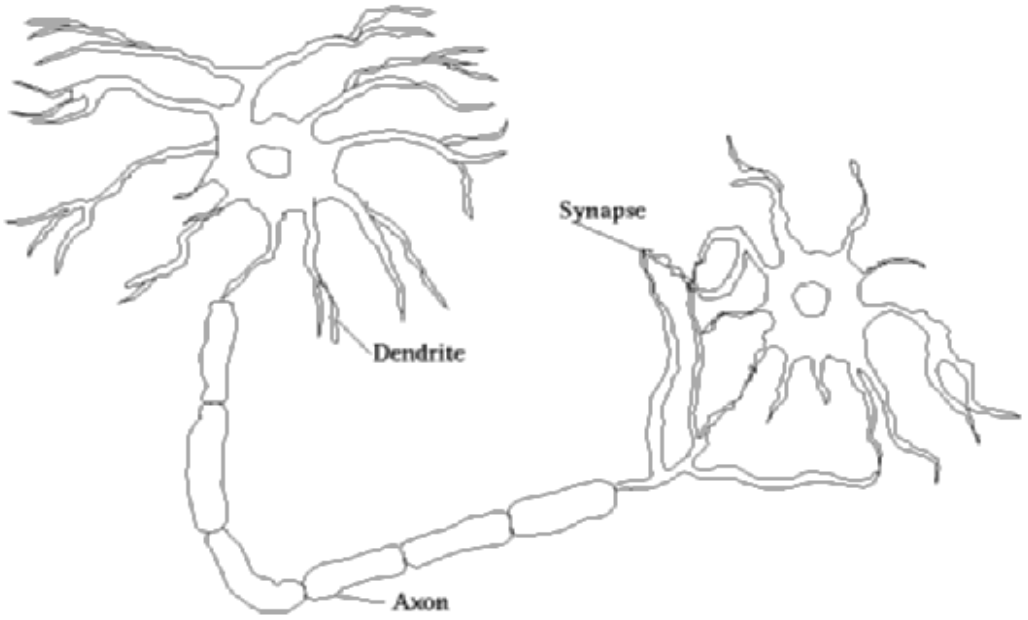
Se estima que el cerebro humano contiene más de cien mil millones (10^11) de neuronas y 10^14 sinapsis en el sistema nervioso humano. Investigaciones sobre la anatomía del cerebro humano concluyen que existen más de 1 000 sinapsis a la entrada y a la salida de cada neurona.

Es importante tomar en cuenta que aunque el tiempo de conmutación de la neurona (unos pocos milisegundos) se encuentra alrededor de un millón de veces menor que en los actuales elementos de las computadoras, ellas tienen una conectividad miles de veces superior que las actuales supercomputadoras.

El objetivo principal de de las redes neuronales de tipo biológico es desarrollar un elemento sintético para verificar las hipótesis que conciernen a los sistemas biológicos.

Las neuronas y las conexiones entre ellas (sinapsis) constituyen la clave para realizar el procesado de la información. Observe la figura:

Figura 3. **Neurona y sus conexiones**



Fuente: http://www.neurocirugia.com/wiki/doku.php?id=neurona. Consulta: 15 de diciembre de 2011.

La mayoría de las neuronas poseen una estructura de árbol llamadas dendritas que reciben señales de entrada provenientes de otras neuronas a través de uniones llamadas sinapsis. Algunas neuronas se comunican solo con las cercanas, mientras que otras se conectan con miles.

Existen tres partes en una neurona:

A. El cuerpo de la neurona,
B. Ramas de extensión llamadas dendritas para recibir las entradas, y
C. Un axón que lleva la salida de la neurona a las desdirías de otras neuronas.

La forma en que dos neuronas interactúan entre sí, no está totalmente comprendida, dependiendo esto además de cada neurona. En general, una neurona envía su salida a otras por su axón. El axón lleva la información por medio de diferencias de potencial, u ondas de corriente, que depende del potencial de la neurona. Este proceso es a menudo modelado como una regla de propagación representada por la función de red u(.) (Descrita más adelante). La neurona recoge las señales por su sinapsis sumando todas las influencias excitadoras e inhibidoras. Si las influencias excitadoras positivas dominan, entonces la neurona da una señal positiva y manda este mensaje a otras neuronas por sus sinapsis de salida. En este sentido la neurona puede ser modelada como una simple función escalón f(.). Como se muestra en la próxima figura, la neurona es activada si la fuerza combinada de la señal de entrada es superior a un cierto nivel, en el caso general el valor de activación de la neurona viene dado por una función de activación f(.)(Descrita más adelante).

2.2. Redes neuronales para aplicaciones concretas

Las ANNs dirigidas a aplicación están en general poco relacionadas a las redes neuronales biológicas. Debido a que el conocimiento que se posee sobre el sistema nervioso en general se encuentra incompleto, se han de definir otras funcionalidades y estructuras de conexión distintas a las vistas desde la perspectiva biológica, es por esto que no son descritas en este documento, por no representar un aporte al propósito de la investigación.

2.3. Taxonomía de las redes neuronales

Toda aplicación de redes neuronales está conformada de dos fases: la fase de aprendizaje o entrenamiento y la fase de prueba. En la fase de entrenamiento, se utilizan conjuntos de datos o patrones de entrenamiento para determinar los pesos (parámetros de diseño) que definen el modelo neuronal. Una vez entrenado este modelo, se usará en la llamada fase de prueba o funcionamiento directo, en la que se procesan los patrones de prueba que constituyen la entrada habitual de la red, analizándose de esta forma las prestaciones definitivas de la red neuronal.

Fase de Prueba: los parámetros de diseño de la red neuronal se han obtenido a partir de los patrones representativos de las entradas, denominados patrones de entrenamiento. Los resultados pueden ser tanto calculados instantáneamente como adaptados iterativamente, según el tipo de red neuronal, y en función de las ecuaciones dinámicas de prueba. Una vez se hayan calculados los pesos de la red, los valores de las neuronas de la última capa, deben ser comparados con la salida deseada para determinar la validez del diseño.

Fase de Aprendizaje: una característica importante de las ANN es su capacidad de aprender. Aprenden por la actualización o cambio de los pesos sinápticos que caracterizan a sus conexiones. Los pesos se adaptan de acuerdo a la información que se extrae de los patrones de entrenamiento nuevos que se van presentando. Normalmente, los pesos óptimos se obtienen optimizando alguna función de energía (minimizando o maximizando). Por ejemplo, un criterio popular en el entrenamiento supervisado es minimizar el valor de salida actual.

Las aplicaciones del mundo real deben abordar dos tipos diferentes de requisitos en el procesado. En uno de los casos, se requiere la prueba en tiempo real pero el entrenamiento ha de realizarse fuera de línea. En otras ocasiones, son requeridos los dos requisitos, el de prueba y el de entrenamiento en tiempo real. Estos dos requisitos implican velocidades de proceso muy diferentes, que afectan a los algoritmos y hardware usados.

Atendiendo al tipo de entrenamiento, una posible taxonomía de las redes neuronales artificiales es:

Tabla I. **Posible taxonomía de las redes neuronales**

Redes Neuronales		
Fijo	**No supervisado**	**Supervisado**
Red de Hamming Red de Hopfield	Mapa de características Aprendizaje competitivo	Basadas en decisión Perceptrón ADALINE (LMS) Perceptrón Multicapa Modelos Temporales Dinámicos Modelos Ocultos de Markov

Fuente: elaboración propia.

2.4. Redes neuronales supervisadas y no supervisadas

Las redes neuronales con clasificadas comúnmente en términos de sus correspondientes algoritmos o métodos de entrenamiento: redes de pesos fijos, redes no supervisadas, y redes de entrenamiento supervisado. Para las redes de pesos fijos no existe ningún tipo de entrenamiento.

2.4.1. Reglas de entrenamiento supervisado

Las redes de entrenamiento supervisado han sido los modelos más desarrollados desde los inicios de estos diseños. La información para el entrenamiento está constituida por varios pares de patrones de entrenamiento de entrada y de salida. El hecho de conocer la salida implica que el entrenamiento es beneficiado por la supervisión de un maestro. Dado un nuevo patrón de entrenamiento, por ejemplo, (m+1)-ésimo, los pesos serán adaptados de la forma ver figura 4.

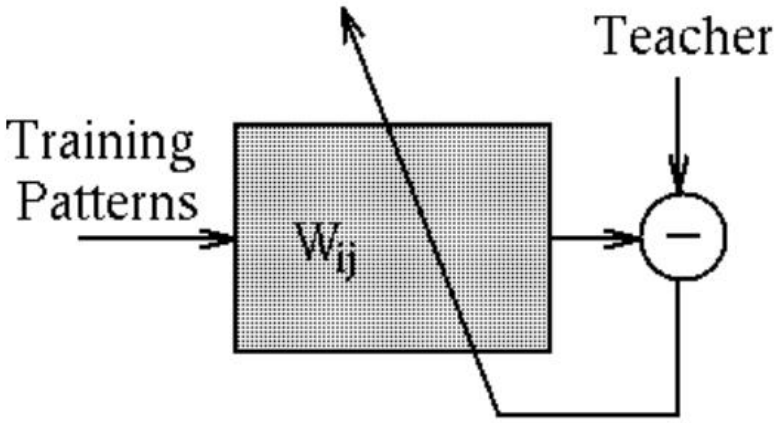
Figura 4. **Ecuación de los pesos sinápticos**

$$w_{ij}^{(n+1)} = w_{ij}^{(n)} + \Delta w_{ij}^{(n)}$$

Fuente: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm. Consulta: 15 de diciembre de 2011.

Se puede ver un diagrama esquemático de un sistema de entrenamiento supervisado en la figura 5:

Figura 5. **Esquema de sistema de entrenamiento supervisado**

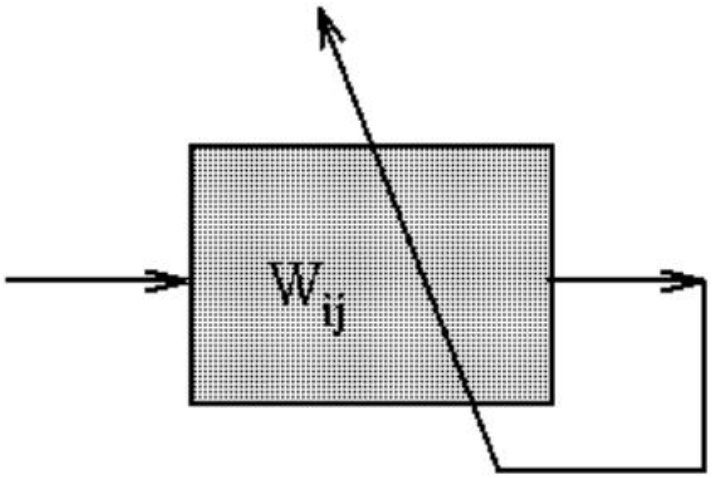


Fuente: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm. Consulta: 15 de diciembre de 2011.

2.4.2. Reglas de entrenamiento no supervisado

Para los modelos de entrenamiento No Supervisado, los conjuntos de datos de entrenamiento consisten sólo en los patrones de entrada. Por lo tanto, la red debe ser entrenada sin el beneficio de un maestro. La red aprende a adaptarse basándose en las experiencias recogidas de los patrones de entrenamiento anteriores. Este es un esquema típico de un sistema No Supervisado:

Figura 6. **Esquema de sistema de aprendizaje no supervisado**



Fuente: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm. Consulta: 15 de diciembre de 2011.

2.5. Funciones de base y activación

Una red neuronal típica puede ser caracterizada por las descripciones funcionales de la red de conexión y la red de activación. Cada célula (unidad de proceso), provee un valor (Ai) a su salida. Este valor es propagado a

través de la red de conexiones unidireccionales hacia otras células de la red. Asociada a cada conexión hay un peso sináptico denominado por {Wij}, que determina el efecto producido por la célula j-pésima sobre la célula i-pésima. Las entradas a la célula i-pésima provenientes de las otras células θi son acumuladas junto con el umbral externo para dar el valor de red U. La forma de hacerlo lo determina matemáticamente la función de base f para dar un valor de activación Ai. La salida final y se puede expresar como una función de la entrada y los pesos.

2.5.1. Función de base (Función de Red)

Para un estudio analítico, las redes de conexión son matemáticamente representadas por la función de base u(w,x), donde w es la matriz de pesos, y x el vector de entrada. La función de base tiene dos formas típicas:

Función Lineal de Base (LBF) es una función de tipo hiperplano. El valor de red es una combinación lineal de las entradas. Esta es una función de primer orden.

Figura 7. **Función lineal de base (LBF)**

$$y = \phi(x, W)$$

Fuente: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm. Consulta: 15 de diciembre de 2011.

Función de base Radial (RBF) es una función de tipo hiperesférico. Esto implica una función de base de segundo orden no lineal. El valor de red representa la distancia a un determinado patrón de referencia.

Figura 8. **Función de base radial (RBF)**

$$u_i(\mathbf{w},\mathbf{x}) = \sqrt{\sum_{j=1}^{n}(x_j - w_{ij})^2}$$

Fuente: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm. Consulta: 15 de diciembre de 2011.

La función de segundo orden se puede extender a otra más general llamada función de base elíptica.

2.5.2. Función de activación (Función de neurona)

El valor de red, expresado por la función de base, u(w,x), será inmediatamente transformada por una función de activación no lineal. Por ejemplo, las funciones de activación más comunes son la función paso, rampa o sigmoidal y gausiana.

Figura 9. **Función sigmoidal**

$$f(u_i) = \frac{1}{1 + e^{-u_i/\sigma}}$$

Fuente: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm. Consulta: 15 de diciembre de 2011.

Figura 10. **Función Gaussiana**

$$f(u_i) = ce^{-u_i^2/\sigma^2}$$

Fuente: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm. Consulta: 15 de ficiembre de 2011.

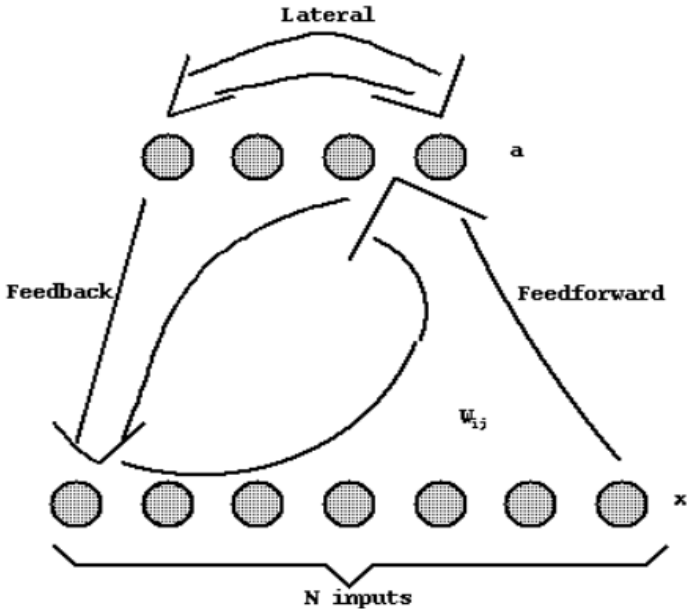
2.6. Estructuras de las redes neuronales artificiales

Los aspectos que más caracterizan a las estructuras son: la estructura de conexión, el tamaño de la red y la elección entre ACON (todas las clases en una red) y OCON (una clase en una red).

2.6.1. Estructuras de conexión de atrás hacia adelante

Una red neuronal es determinada por la neurona y la matriz de pesos. El comportamiento de la red neuronal depende en gran parte del comportamiento de la matriz de pesos. Hay tres tipos de capas de neuronas: la de entrada, las ocultas y la de salida. Entre dos capas de neuronas existe una red de pesos de conexión, que puede ser de los siguientes tipos: hacia delante, hacia atrás, lateral y de retardo, tal como puede se puede ver en la siguiente figura:

Figura 11. **Red de pesos de conexión**



Fuente: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm. Consulta: 15 de diciembre de 2011.

A. Conexiones hacia delante: para todos los modelos neuronales, los datos de las neuronas ubicadas en una capa inferior se propagan hacia las neuronas de la capa superior por medio de las redes de conexión hacia adelante.

B. Conexiones hacia atrás: estas conexiones propagan los datos de las neuronas de una capa superior a otras de la capa inferior.

C. Conexiones laterales. Un ejemplo típico de este tipo es el circuito el ganador toma todo (winner-takes-all), que cumple un papel importante en la elección del ganador, este tipo es descrito más adelante como parte de las redes de aprendizaje competitivo.

D. Conexiones con retardo: los elementos de retardo son incorporados en las conexiones para implementar modelos dinámicos y temporales, es decir, modelos que precisan de memoria.

2.6.2. Tamaño de las redes neuronales

El tamaño de las redes depende del número de capas y del número de neuronas ocultas por capa.

Número de capas: en una red multicapa, hay una o más capas de neuronas ocultas entre la entrada y la salida. El número de capas se cuenta a menudo a partir del número de capas de pesos (en vez de las capas de neuronas).

Número de unidades ocultas: El número de unidades ocultas está directamente relacionado con las capacidades de la red. Para que el comportamiento de la red sea correcto (esto es, generalización), se tiene que determinar apropiadamente el número de neuronas de la capa oculta.

2.6.3. Aproximaciones ACON frente a OCON

Dos posibles tipos de arquitectura son All-Class-in-One-Network (ACON), esto es, todas las clases en una red y One-Class-in-One-Network (OCON), esto es, una red para cada clase. En la aproximación ACON, todas las clases son reconocidas dentro de una única súper red. En algunos casos es ventajoso descomponer esta macro red en varias subredes más pequeñas. Por ejemplo, una red de 36 salidas se puede descomponer en 12 subredes, cada una responsable de tres salidas. La descomposición más extrema es la llamada OCON, donde una subred se dedica para una sola clase. Aunque el número de

subredes en la estructura OCON es relativamente largo, cada subred individual tiene un tamaño menor que la red ACON. Esto se puede explicar con las siguientes figuras,

Figura 12. **Figuras ACON y OCON**

Fuente: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm. Consulta: 15 de diciembre de 2011.

La red entera, se divide en ver figura 13

Figura 13. **Subredes**

Fuente: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm. Consulta: 15 de diciembre de 2011

Por conveniencia, supóngase que todas las redes tienen el mismo tamaño, por ejemplo k. El número de unidades ocultas de las macro-red ACON se denota por K. Obviamente, k << K. Las dos estructuras difieren claramente en tamaño y rapidez, esto es, en el número total de pesos

sinápticos y en el tiempo de entrenamiento. Sean los vectores de entrada y salida de dimensiones n y N respectivamente. El número total de pesos sinápticos es para la estructura ACON (N+n) * K. De la misma forma, el número para la estructura OCON es N * (n+1) * k approx. N*n * k . Los dos casos extremos son analizados a continuación. Cuando N es relativamente pequeña (comparado con n), la estructura ACON podría tener el mismo número de pesos o algo menos que la OCON. Si N es muy grande, entonces la OCON podría tener una mayor ventaja en términos del tamaño de la red. Además, la OCON parece ser que aventaja a la ACON en la rapidez de reconocimiento y entrenamiento cuando el número de clases es grande.

En la estructura ACON, la única macro-red tiene que satisfacer todas estas clases, así que el número de unidades ocultas K ha de ser muy grande.

2.7. Modelos no supervisados

La capacidad de clasificación de la red neuronal depende del valor de los pesos sinápticos, que pueden ser preestablecidos o entrenados adaptativamente mediante mecanismos de aprendizaje. Las NNs se pueden clasificar, atendiendo a como sean entrenados los pesos sinápticos, en dos grandes categorías: estas son los modelos supervisados y los no supervisados.

Una clase de modelos de entrenamiento no supervisado son las redes de pesos fijos. Un ejemplo son las redes de Memoria Asociativa, que se usan para obtener patrones originales libres de ruido a partir de señales incompletas o distorsionadas. La principal característica de las redes asociativas de pesos fijos es que sus pesos son preestablecidos y precalculados.

Los modelos de pesos fijos tienen aplicaciones limitadas ya que no se pueden adaptar a ambientes cambiantes. Hay otra variedad de redes no supervisadas, llamadas Redes de Aprendizaje Competitivo, cuyos pesos se adaptan de acuerdo con reglas de aprendizaje no supervisadas. Estas redes pueden aprender en ausencia de un maestro. En otras palabras, el entrenamiento se basa únicamente en la información de los patrones de entrada. La clase de redes de aprendizaje competitivo se componen, por ejemplo, de la Red de Auto organización.

2.8. Modelos supervisados

Las ANNs de entrenamiento supervisado constituyen la línea fundamental de desarrollo en este campo. Algunos ejemplos bien conocidos de las primeras redes son red perceptrón, ADALINE/MADALINE, y varias redes multicapa. En el entrenamiento supervisado hay dos fases a realizar: fase de prueba and fase de entrenamiento.

En el entrenamiento supervisado, los patrones de entrenamiento se dan en forma de pares de entrada/maestro,

Figura 14. **Patrones de entrenamiento supervisado**

$$[\mathcal{X}, \mathcal{T}] = \{[\mathbf{x}_1, \mathbf{t}_1], [\mathbf{x}_2, \mathbf{t}_2], \cdots, [\mathbf{x}_M, \mathbf{t}_M]\},$$

Fuente: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm. Consulta: 15 de diciembre de 2011.

En la figura 14, donde M es el número de pares de entrenamiento. Dependiendo de la naturaleza de la información del maestro, hay dos aproximaciones al entrenamiento supervisado. Uno se basa en la corrección a partir de una decisión y la otra se basa en la optimización de un criterio de coste. De la última, la aproximación del error cuadrático medio es el más importante. Las formulaciones de decisión y aproximación difieren en la información que tienen los maestros y la forma de usarla.

Son entonces, las ANN son un modelo de aprendizaje artificial muy apropiado para incorporarlas con la teoría de la personalidad de Albert Bandura descrita en el capítulo anterior, a lo largo de toda la investigación se pretende ir detallando mas al respecto de cómo sería la implementación correcta del modelo de ANN más apropiado para el propósito de este proyecto.

3. PERCEPCIÓN DE PROFUNDIDAD Y APRENDIZAJE POR IMITACIÓN

Dentro del aprendizaje por imitación existen dos aspectos importantes que funcionan como pautas para lograr el aprendizaje, los cuales son: la identificación de un modelo a imitar teniendo la conciencia que es un modelo de sí mismo, y el conjunto de acciones que el modelo realiza que se quieren imitar. Si se quiere implementar esto a nivel de software conviene comprender conceptualmente como funciona esto dentro del cerebro humano.

Para que el cerebro humano identifique un modelo de sí mismo como ser humano, es necesario que primero tenga abstraído el concepto de lo que es un ser humano, este concepto se va formando en las etapas muy tempranas de la infancia del individuo, a través de lo que recibe de sus sentidos, la vista como tal, proporciona una cantidad increíble de información para poder abstraer conceptos de cada objeto dentro del cerebro, por ejemplo si un individuo nunca en su vida ha visto una copa para beber, si se le intenta explicar el concepto de una copa sin hacer referencia a la forma física de la copa, será un tanto más difícil transmitirle ese concepto, pero sí de manera contraria se le muestra al individuo un conjunto de copas, este podrá abstraer el concepto mucho más rápido y fácilmente, que tomara unos cuantos segundos transmitirle el concepto.

Figura 15. **Conjunto de copas**

Fuente: http://idecoracion.net. Consulta: 15 de diciembre de 2011.

El cerebro, al ver la copa, automáticamente almacena la información de la forma física de la copa, y almacena esta información de manera tridimensional, gracias a la percepción de profundidad que el cerebro posee al interpretar la información que llega de sus ojos. Se sabe que los ojos envían 72 GB de información por segundo al cerebro humano, la cual el cerebro procesa para conocer la forma y posición tridimensional dentro del entorno que lo roda constantemente. ¿Pero qué pasaría si no se tuviera la percepción de profundidad?, sería muy difícil diferenciar la forma y posición de los objetos y abstraer un buen concepto de ellos, como por ejemplo al ver la siguiente figura:

Figura 16. **Figura bidimensional**

Fuente: http://aniak.galeon.com. Consulta: 15 de diciembre de 2011.

Al ver la imagen anterior tanto se puede decir que tiene la forma de una copa, pero de la misma manera tiene la forma de dos rostros que se ven uno frente al otro, esto es porque aquí no se tiene percepción de profundidad por qué la imagen esta en dos dimensiones.

3.1. El niño real

Cuando un niño va creciendo, va formando conceptos de cada cosa, por eso es común ver que un niño constantemente puede estar preguntando ¿Qué es eso? al ver un objeto que no conoce, después de responderle por ejemplo: Esa es una pelota, lo más probable es que el niño siga con lo que estaba haciendo, pues ya almacenó la imagen tridimensional en su cerebro y el nombre del objeto, de manera semejante es común que un niño de más edad pregunte también ¿y para qué sirve?, porque su cerebro ya sabe que los objetos pueden tener un propósito de existencia, pero un niño pequeño le bastara solo con la imagen tridimensional y el nombre del objeto.

Una vez el individuo tiene abstraído el concepto de lo que es un ser humano, puede auto clasificarse como ser humano y ver a los demás como posibles modelos a imitar. Lo cual ya permite realizar un experimento como el que hizo Albert Bandura con el muñeco bobo (descrito en el primer capítulo de este documento).

Por ejemplo, cuando una persona ve patear un balón a otra, la información que recibe la interpreta como un conjunto de posiciones sucesivas de cada parte del cuerpo de la otra persona en relación con la pelota y el entorno que los rodea, después asocia estos movimientos a sus propias extremidades y decide si va a intentar o no imitar el comportamiento, aquí también se ve la percepción de profundidad como un eslabón importante para lograr el aprendizaje por imitación.

Pero como poder simular esto a nivel de software, lograr que un software aprenda mediante imitación, primero es necesario colocar el software en un entorno, en un ambiente donde pueda identificar a un modelo de sí mismo, es decir, tener conciencia de la semejanza que existe, hacer que el modelo realice un conjunto de acciones que el software perciba, y que asocie a lo que posee el software como herramientas propias, que serian en este caso sus extremidades. Por estas razones que un entorno virtual es ideal para poder implementar claramente estos procesos a nivel de software.

3.2. El niño virtual

Uno de los propósitos de este documento, es hacer conciencia de la gran cantidad de posibilidades que un entorno tridimensional ofrece para la investigación de la inteligencia artificial sin la necesidad de sensores físicos.

Por ejemplo tal vez en un futuro no tan lejano sea posible la implementación de un niño virtual, es decir, que un modelo de un niño tenga las habilidades de percepción de profundidad dentro de un entorno simulado por computadora, asociación de imágenes tridimensionales de los objetos y sus nombres, así como el aprendizaje por imitación, por ejemplo: al iniciar el programa de aprendizaje, un modelo de niño virtual aparece en un cuarto vacío sin puertas ni ventanas, solo con una caja, el niño virtual deberá ser capaz de iniciar un proceso de aprendizaje que le obligue a preguntar que es el objeto que ve y que también pueda percibirlo en tres dimensiones, el usuario por medio de alguna interfaz pueda comunicarle al niño el nombre del objeto, de manera consecuente se le seguiría mostrando nuevos objetos, hasta que se le muestra un modelo idéntico al niño, nuevamente el niño deberá preguntar que es, se le comunica que es un niño, el usuario deberá realizar acciones con el nuevo niño como caminar, saltar, sentarse, acostarse, etc.

En donde entraría en acción los modelos de aprendizaje artificial por imitación, luego que el niño aprenda tantos movimientos como sea posible, el siguiente paso sería agregar una puerta al cuarto, y que el usuario use el niño a imitar hacia la puerta para que salga del cuarto y pase a otro, y que el niño virtual imite nuevamente dicho comportamiento, y así de manera sucesiva, hasta que el niño tenga una inmensa cantidad de nuevo aprendizaje en donde solo faltaría estudiar como proveerle de la facultad de tomar decisiones por sí mismo para que su comportamiento sea sumamente semejante en cuanto a lo que respecta a la conducta básica de un ser humano.

Figura 17. **El niño virtual**

Fuente: http://aniak.galeon.com. Consulta: 15 de diciembre de 2011.

3.3. El entorno real y el entorno virtual

Como se puede ver, uno de los factores muy importantes para el desarrollo de nuevo aprendizaje dentro del cerebro y la personalidad viene del entorno que rodea al individuo, y para ser más exactos, viene de lo que sus sentidos perciben de ese entorno o ambiente.

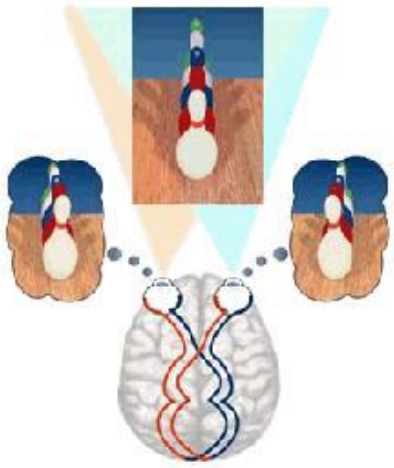
Dentro del entorno real que rodea al ser humano existen una gran cantidad de fenómenos, situaciones, condiciones, objetos, colores, ondas, partículas, etc. De las cuales una gran cantidad son percibidas por el cerebro, y que producen cambios dentro del cerebro. Dentro de los avances tecnológicos hoy en día, es difícil y costoso construir sensores que brinden la cantidad de información y precisión que los sentidos humanos proveen, para poder realizar investigaciones del aprendizaje por imitación artificial sería sumamente costoso realizarlo dentro del entorno real, pues como se ha podido ver, la percepción de profundidad es clave para este proceso, y obtener

sensores que brindarán esa información, excedería el presupuesto de muchos investigadores.

Para entender lo difícil de implementar esta percepción de profundidad debe señalarse una breve descripción del proceso.

Según investigaciones, la percepción de profundidad es procesada en el cerebro a partir de varios factores, dentro de los cuales cabe destacar el análisis de dos imágenes (una de cada ojo) de un mismo objeto, pero en diferente ángulo, esto es fácil demostrarlo extendiendo el pulgar al frente y cerrar un ojo, y luego abrirlo y cerrar el otro, claramente podrá verse las diferencias de ángulo con las imágenes que están más lejanas al propio pulgar.

Figura 18. **Proceso binocular**

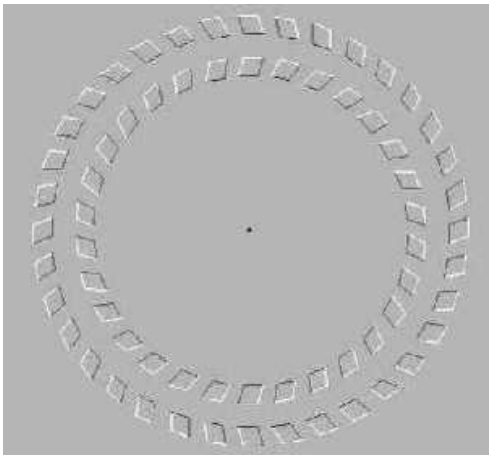


Fuente: http://aniak.galeon.com. Consulta: 15 de diciembre de 2011

La imagen anterior ilustra el proceso binocular para forma parte de la percepción de profundidad.

El proceso de percepción de profundidad es tan complejo a nivel biológico que resulta sumamente difícil poder implementarlo o costearlo a nivel de robótica para la investigación del aprendizaje artificial por imitación. Un ejemplo de las implicaciones de los procesos de percepción visual es la siguiente imagen, que al ver fijamente el punto del centro y alejar y acercar la cabeza produce un efecto curioso debido a los distintos ángulos que se reciben para cada figura.

Figura 19. **Prueba de percepción**



Fuente: http://aniak.galeon.com. Consulta: 15 de diciembre de 2011.

Por estas razones es necesario trasladar el campo de investigación a un entorno más flexible pero que brinde todas las posibilidades que se necesitan o incluso mayores.

Dentro de un entorno virtual, es decir, en un mundo tridimensional generado por computadora (con las tecnologías actuales), es posible generar modelos físicos prácticamente de casi todos los objetos que rodean a un ser humano en la realidad, sin embargo, estos mundos virtuales carecen de las

propiedades físicas inherentes del mundo real, por lo que es necesario implementarlas para obtener una simulación más acercada.

Incluso en estos mundos virtuales es posible construir modelos humanoides a los cuales se les puede dar propiedades que en el mundo físico no existen, es como por ejemplo la facultad de poder volar sin necesidad de ningún artefacto ajeno al humanoide, aunque el humanoide realmente no está volando, solo cambia su posición dentro del entorno virtual, es decir, es una simulación.

4. EL MODELO DE REDES NEURONALES ESPECULARES

Después de conocer los factores más influyentes para lograr el aprendizaje por imitación dentro de un contexto psicológico, ¿Cómo trasladar esto a nivel de software?, para realizar esto, primero se necesita decidir qué modelo de red neuronal se acomoda más a lo que se requiere que realice el software. A continuación se presenta la descripción de un experimento que provee un fuerte aporte a esta investigación.

El Dr. Giacomo Rizzolatti, profesor de fisiología humana y neurólogo en la Universidad de Parma realizó un experimento muy curioso junto con su equipo de trabajo en la corteza cerebral de los monos. El experimento reveló de la existencia de neuronas que se activaban no sólo cuando el simio estaba realizando una determinada acción motora, sino también cuando el animal observaba la misma acción pero realizada por otro animal generalmente de la misma especie.

A estas neuronas se les llamó neuronas especulares. Tal experimento pasó a ser un hito importante en el campo de la neurociencia y la psicología. El descubrimiento de estas neuronas especulares y su posible papel en el aprendizaje por imitación y en fenómenos como la empatía ayudarán a explicar y conocer muchas capacidades mentales que permanecían misteriosas y eran inaccesibles a la experimentación.

Giacomo Rizzolatti y su equipo haciendo registros de la actividad neuronal de los monos observaron que ciertas neuronas en la corteza premotora cerebral se activaban cuando el mono ejecutaba un determinado movimiento con un brazo o mano, como, por ejemplo, tomar un trozo de fruta, empujar algún objeto

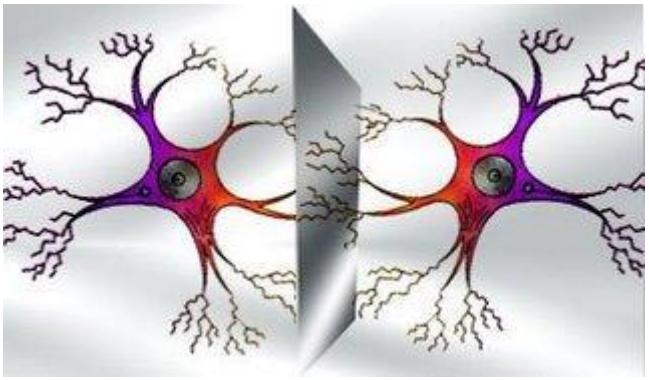
o llevarse un cacahuete a la boca, además que para cada acción diferente se activaban diferentes neuronas. Es decir,, la activación de una neurona determinada era dependiente de la acción que se ejecutaba. Estas neuronas ya habían sido descubiertas y se les considera como neuronas motoras superiores o neuronas de «comandos motores», este tipo de células se activan específicamente con movimientos complejos o secuencias complejas de movimientos. Más adelante se describirá como estas neuronas motoras formaran parte del modelo de red neuronal artificial a utilizar para esta investigación.

Una vez llegado a este punto es necesario explicar, de forma muy sencilla, cómo se generan los movimientos voluntarios, en otras palabras, los movimientos dirigidos con un propósito (por ejemplo, tomar un trozo de fruta), en las diferentes áreas de la corteza cerebral. Los primeros estímulos (que serían la observación de la fruta), son procesados en las cortezas sensoriales (principalmente en la corteza visual del cerebro, donde cabe notar la importancia de la percepción de profundidad para ver el entorno en tres dimensiones), seguidamente la información es transmitida a las cortezas asociativas del cerebro, donde se decide la acción a ejecutar. Esa intención en realizar la acción es enviada a las regiones premotoras, donde segenera la secuencia tan complicada de comandos producirán el movimiento final deseado, esdecir, alcanzar y tomar la fruta (aún, desde la corteza premotora, las órdenes tienen que ser enviadas a la corteza motora y finalmente a las motoneuronas de la médula, que son las que antevienen en los músculos del brazo y de la mano, siendo responsables finales para realizar el movimiento).

Así las neuronas motoras superiores codifican un determinado movimiento. Para esta investigación las neuronas motoras codificarán los movimientos en las extremidades del modelo virtual humano.

Lo interesante del experimento de Rizzolatti es el hecho de que las mismas neuronas que se activaban cuando el mono realizaba la acción, se activaban cuando el mono veía a otro realizar dicha acción, sin que el primer mono tuviera que realizar alguna actividad física, es decir, que la activación de estas neuronas especulares también es específica de la tarea que se está observando.

Figura 20. **Redes neuronales especulares**



Fuente: http://aniak.galeon.com. Consulta: 15 de diciembre de 2011.

El descubrimiento de las neuronas especulares en la corteza promotora cerebral apunta a que la simple observación de un movimiento específico en otro individuo es suficiente para genera una simulación mental del movimiento que se está observando, es decir, que mientras se ve un movimiento de otro individuo se va construyendo una copia mental del movimiento que posteriormente será reproducida físicamente aunque probablemente si el movimiento es muy complejo, esta copia no será precisamente idéntica, pero se procederá a corregir hasta conseguir el nivel deseado de similitud del movimiento (imitación). De esta manera, la imitación puede considerarse una tendencia automática de la respuesta, aunque ésta se encuentre normalmente reprimida.

La comprensión de las redes neuronales especulares es aún muy corta, por su muy reciente hallazgo, pero su concepto es esencial para desarrollar su modelo a nivel de software dentro de un ambiente tridimensional. La existencia de neuronas y/o sistemas especulares podría dar explicación no solamente a la facilidad para el aprendizaje de nuevos movimientos o la tendencia natural animal o humana a la imitación de los gestos u otros movimientos, así como también la capacidad de hacer propia la forma en que sienten otras personas y de compartir sus sentimientos (empatía), lo cual llevaría a una mejor comprensión de su comportamiento o de su forma de tomar decisiones, que al final existe también la posibilidad de simularlo por modelos de empatía artificial en agentes sociales.

4.1. Perceptrón multicapa como modelo especular

Dentro de los modelos existentes a utilizar para implementar un sistema de redes neuronales artificiales especulares, el modelo más acercado es el Perceptrón multicapa. A continuación, se hace una descripción de en lo que consiste dicho modelo y el porqué de su elección.

Primero es necesario mencionar que el modelo perceptrón multicapa es un modelo de aprendizaje supervisado, es decir, que necesitan un conjunto de datos de entrada previamente clasificado o cuya respuesta objetivo se conoce, para esta investigación seria entonces la secuencia de movimiento correcta que imite satisfactoriamente el movimiento del modelo humanoide en tres dimensiones.

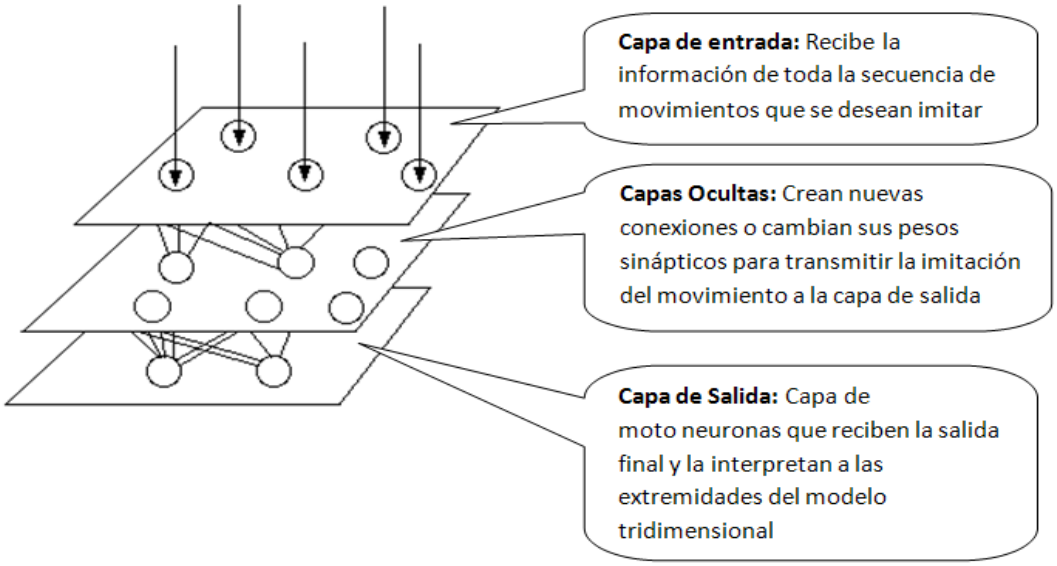
El perceptrón multicapa es una red neuronal artificial formada por múltiples capas, esto le permite resolver problemas que no sean linealmente separables. El perceptrón multicapa puede ser totalmente o localmente. En el primer caso

cada salida de una neurona de la capa i es entrada de todas las neuronas de la capa i+1, mientras que el segundo, cada neurona de la capa i es entrada de una serie de neuronas (región) de la capa i+1.

Las capas pueden clasificarse en tres tipos:

- Capa de entrada: Constituida por aquellas neuronas que introducen los patrones de entrada en la red. En estas neuronas no se produce procesamiento.
- Capas ocultas: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y las salidas pasan a neuronas de capas posteriores.
- Capa de salida: Neuronas cuyos valores de salida se corresponden con las salidas de toda la red.

Figura 21. **Funcionamiento de perceptrón**

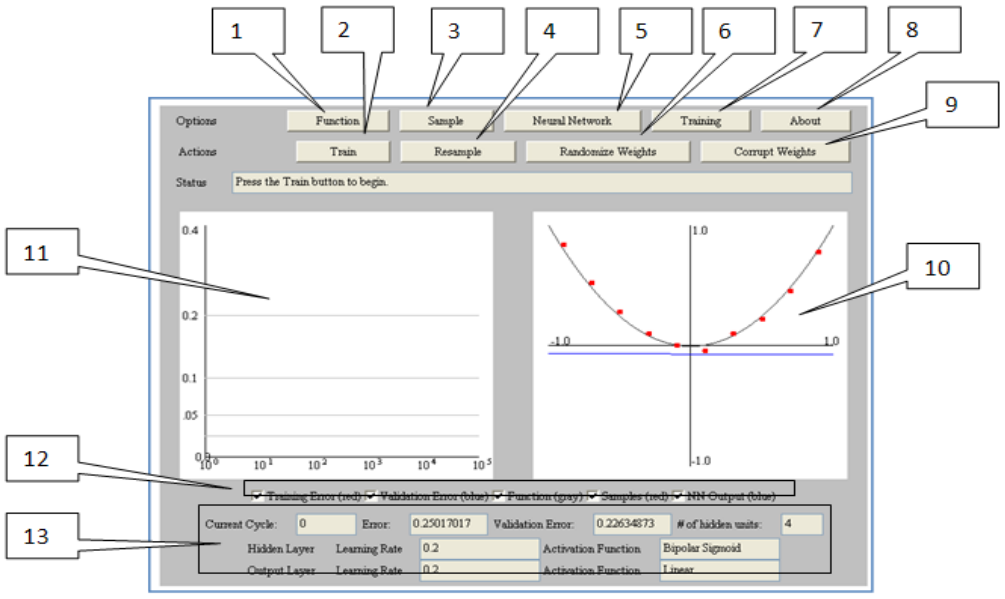


Fuente: elaboración propia.

La propagación hacia atrás (también conocido como retropropagación del error o regla delta generalizada), es un algoritmo utilizado en el entrenamiento de estas redes, por ello, el perceptrón multicapa también es conocido como red de retropropagación.

Como parte de los objetivos de esta investigación, es que la red neuronal vaya mejorando los movimientos del modelo humanoide, se requiere un modelo que conforme vaya siendo entrenado perfeccione dichos movimientos, por eso el modelo Perceptrón multicapa es el más adecuado para la tarea, por ejemplo, a continuación se muestra un applet que implementa una red neuronal de tipo Perceptrón multicapa que conforme se va entrenando, se va acercando a la gráfica de una ecuación matemática, así como también la descripción de la funcionalidad y áreas de configuración de dicha aplicación.

Figura 22. **Herramienta ejemplo de redes neuronales**



Fuente: elaboración propia.

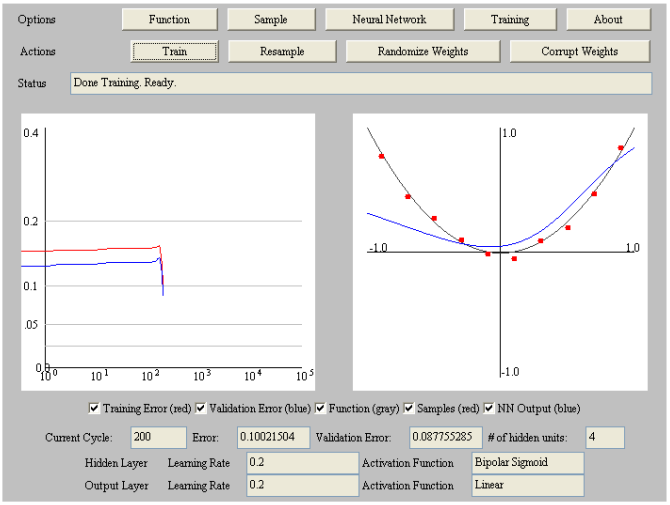
Tabla II. **Descripción de figura 22**

No.	Descripción
1	El Botón **Function** permite especificar el tipo de función matemática que se va a
2	El Botón **Train** ejecuta una iteración de entrenamiento sobre la red
3	El Botón **Sample** permite especificar la información de las variables de entrada que son los puntos rojos sobre la curva, por ejemplo cuantos puntos se van
4	El Botón **Resample** obtiene nuevos puntos rojos sobre la curva
5	El Botón **Neural Network** permite editar las propiedades de la red neuronal como la
6	El Botón **Random Weights** inicializa los pesos en valores random.
7	El Botón **Training** edita las propiedades del entrenamiento como el número de
8	El Botón **About** muestra información del autor.
9	El Botón **Corrupt Weights** inicializa los pesos con valores alterados según indique
10	El área del **plano cartesiano** muestra tanto la curva que se tiene que aprender como la
11	El área de **gráficos de aprendizaje** muestra gráficamente el error que se presenta conforme se va aprendiendo, así de manera consiguiente la curva
12	En esta área se selecciona que se desea ver en las áreas de gráficos, como los puntos rojos, la curva que se quiere aprender, la curva de la red neuronal, y
13	En esta área muestra el estado de todas las variables implicadas en el aprendizaje, como cantidad de capas de la red, la iteración actual, los errores,

Fuente: elaboración propia.

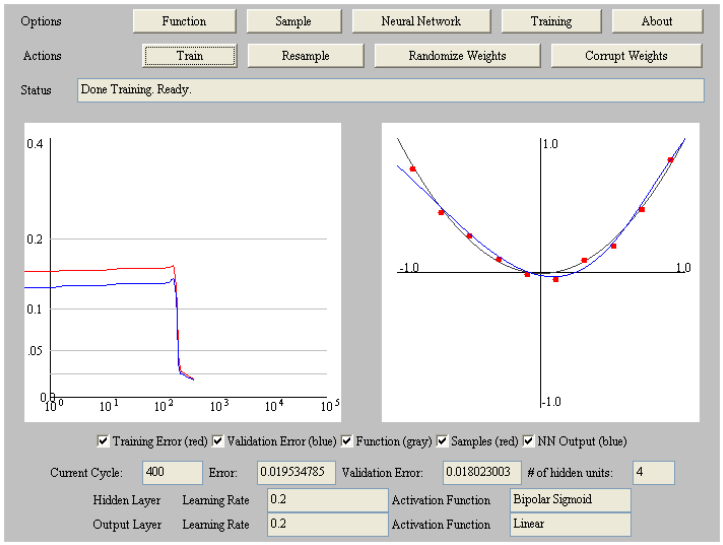
Conforme se da clic en el botón Train la red va acercándose más a la función cuadrática que tiene como fin imitar.

Figura 23. **Entrenamiento de red**



Fuente: elaboración propia.

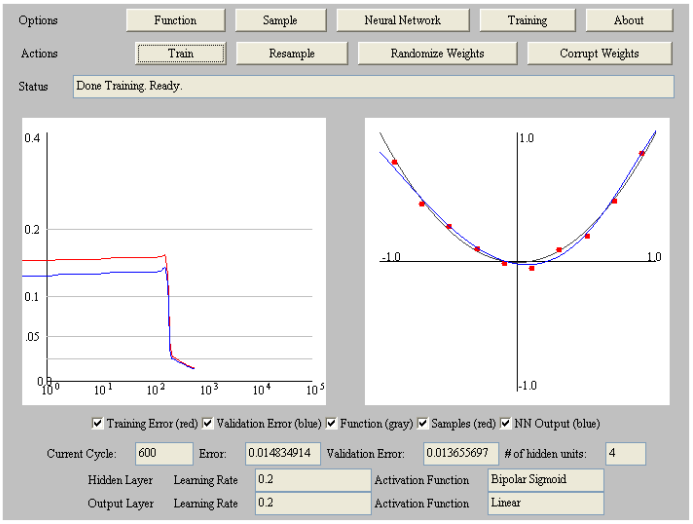
Figura 24. **Entrenamiento de red 2**



Fuente: elaboración propia.

Se sigue entrenando a la red neuronal hasta que alcanza una figura sumamente acercada a la función cuadrática.

Figura 25. **Red neuronal entrenada**



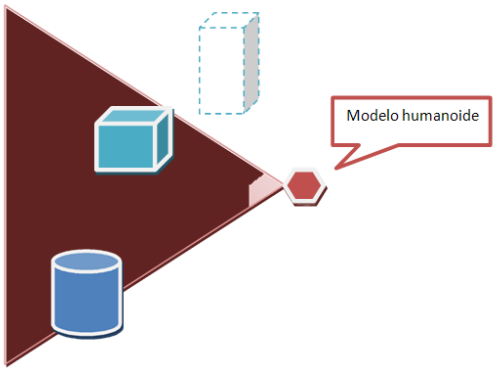
Fuente: elaboración propia.

Este comportamiento encaja perfectamente con el objetivo de imitar una secuencia de movimientos a través de las motoneuronas de cada de salida de la red neuronal.

Del ejemplo anterior se puede ver que el resultado deseado es la parábola de la función cuadrática, que la red neuronal basada en el modelo Perceptrón imitó con un grado de ajuste increíblemente bueno, la diferencia con el modelo de red neuronal especular radica principalmente que lo que se busca imitar es una secuencia de movimientos en un entorno tridimensional, para lo cual se necesita un grafo con coordenadas en tres dimensiones como variables de entrada. Esto se explica a continuación.

Para que el modelo humanoide tenga una percepción de profundidad muy acercada a la del ser humano, tendría que dársele información de la posición de cada objeto y/o vértices de dicho objeto si este cae dentro de un ángulo de visión del modelo humanoide. Como se ilustra en la siguiente figura:

Figura 26. **Ángulo de visión del modelo**



Fuente: elaboración propia.

Es decir, que el modelo humanoide deberá tener un ángulo de visión, un rango de espacio que puede ver y otro que no, así el objeto punteado en la imagen no sería visible paral modelo humanoide, este tendría que rotar sobre su eje vertical para poder visualizar ese objeto, esto además de proporcionar una simulación más acercada a la realidad, reduce la cantidad de información que el software inteligente deberá procesar, sin embargo, el objetivo principal de este proyecto no es simular el proceso de visión sensorial del ser humano, sino analizar cómo sería un software que aprenda por imitación, por esto se puede reducir la visión del modelo humanoide a solo los vértices de los objetos. En consecuencia el modelo humanoide vería a su homologo tridimensional con la siguiente figura:

Figura 27. **Figura homologo**

Fuente: elaboración propia.

Según se puede apreciar la imagen anterior encaja perfectamente con el concepto de un grafo, pero para este caso sería un grafo en tres dimensiones por lo que un movimiento como el que se ilustra a continuación seria una secuencia de coordenadas en x, y, z por nodo del modelo a imitar que tendría entonces el papel de las variables o datos de entrada de la red neuronal.

Figura 28. **Figura homologo varias posiciones**

Fuente: elaboración propia.

Las motoneuronas entonces deben estar conectadas al grafo de extremidades que representa internamente el modelo humanoide que posee inteligencia artificial, para que por cada nuevo entrenamiento se ejecuten los movimientos generados por la red neuronal hasta que se logre imitar con un buen grado de ajuste dichos movimientos.

El formato de las variables de entrada y salida y la forma en que se digitalizan se describen en el análisis formal de esta investigación. Así como también las especificaciones propias que identificarían al modelo de red neuronal especular.

4.2. El entorno HAVOK

Dentro de las herramientas analizadas para la implementación del software del que se hace énfasis en este trabajo de investigación, se ha seleccionado el motor físico Havok.

Havok (Havok Physics) es un motor físico middleware desarrollado por una compañía irlandesa con el mismo nombre, que por medio de simulación dinámica permite recrear interacciones de carácter físico en videojuegos y otras aplicaciones multimedia. Recientemente este motor físico fue adquirido por Intel y liberado bajo una licencia que permite su libre uso para fines no comerciales o educativos.

La aparición de Havok fue en la Gamers Developer Conference en el año 2000, en la versión 1.0, que para hoy su versión más actual es la 5.0 resultante en septiembre de 2007.

Su código fuente está basado en C/C++ por lo posee compatibilidad con cualquier sistema con un compilador C/C++ válido. Este motor físico sido trasladado a Windows, Unix, MacOs X, así como a plataformas de videojuegos por consolas Xbox, Xbox360, Nintendo GameCube, Playstation2, Playstation3, PlaystationPortable (PSP), y Nintendo Wii.

Figura 29. **Videojuegos soportados por HAVOK**

Fuente: Documentación descargable en http://software.intel.com/sites/havok/
Consulta: 15 de diciembre de 2011.

La facilidad de utilización y versatilidad de sus fuentes, así como sus posibilidades le permitieron a este motor físico ser aceptado rápidamente en la industria del videojuego y licenciado para gran mayoría de videojuegos que requerían un motor gráfico, consiguiendo ser percibido prácticamente en la única alternativa, antes de la aparición de su competidor más directo, Ageia PhysX.

La aceptación de Havok es tal que su implementación en construcción de videojuegos a nivel comercial forma parte de muchos títulos del genero conocidos hoy en día, entre los cuales figuran: Age of Empires III, Counter-Strike: Source, Halo 2, Halo 3, Medal of Honor, Spider- Man 2, StarCraft II, The Matrix: Path of Neo, Tom Clancy's Splinter Cell: Chaos Theory.

Entre las posibilidades que ofrece Havok, las más significativas son:

- Físicas rag-doll. Retomar el control del personaje una vez aplicadas las implicaciones físicas, de una manera más o menos realista. (Que para el caso de esta investigación posiblemente sea de poca utilidad)

- Sistema de físicas continúas. Permite que el motor físico actúe de forma permanente, sin estar sujeto a eventos predeterminados.(sumamente útil para cuando el aprendizajepor imitación requiere interacción con otros objetos)

- Portabilidad y facilidad de uso y, debido a estar basado en C/C++.

- Buen rendimiento debido a bajo consumo en memoria.

Dentro de las razones de elección para esta herramienta, son dos particularmente:

- La eliminación de tiempo a invertir en programar el entorno tridimensional con las implicaciones físicas que existen en el mundo real, pues Havok se encarga de detectar colisiones, movimientos resultantes, efectos de la gravedad, etc.

- La presencia tan fuerte de la herramienta a nivel industrial y comercial, lo cual lo hace una herramienta sumamente confiable y de alta calidad.

Para poder ver más información respecto de esta herramienta se puede consultar el anexo correspondiente adjunto en este documento.

5. ANÁLISIS DE VARIABLES DE ENTRADA

El conjunto de variables de entrada del modelo de red neuronal especular como se vio en el capítulo anterior está formado básicamente por las coordenadas cartesianas en tres ejes para cada vértice, sin embargo, el cerebro humano para realizar aprendizaje por imitación realiza una toma de variables de entrada mucho más basta y complicada, por ejemplo, primeramente se debe tomar las coordenadas cartesianas de cada vértice a modo de secuencia, es decir, obteniendo un vector de posiciones x, y, z por cada vértice para poder representar la secuencia del movimiento resultante, por sobre esto el cerebro también toma el tiempo que le tomó al otro ser humano por ejemplo, terminar toda la secuencia total, se toman también variables como la fuerza que se realizó en cada movimiento (de la que se percibe a simple vista), pero una variable de entrada sumamente significativa para el aprendizaje, es el resultado de dicha acción, por ejemplo que al lanzar un balón, este caiga dentro de una portería de fútbol o que al pedalear en una bicicleta se consiga avanzar sin caerse, es decir, el resultado final que obtuvo el otro ser humano sería entonces el indicador contra el cual comparar si se produjo o no aprendizaje, si se logró reproducir la acción con el mismo resultado.

5.1. El robot maestro y el robot discípulo

Básicamente el escenario en donde se ejecutara el software consiste en involucrar dos modelos humanoides, donde uno aprende del otro, utilizando una red neuronal.

De forma más detallada, el robot maestro es el que realiza ciertas acciones de acuerdo a instrucciones enviadas por el usuario, por ejemplo el usuario tendrá varias opciones a realizar con el modelo maestro, por ejemplo, saludar , caminar, correr , saltar, etc.

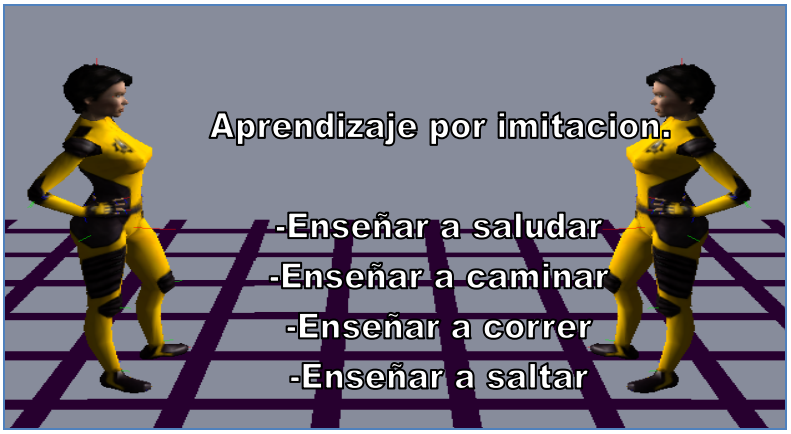
El robot discípulo es el que va a reconocer al robot maestro como el modelo a imitar, y estará atento a los movimientos que este realice.

Es necesario especificar cómo poner a disposición estas secuencias de movimientos desde el robot maestro.

5.1.1. El robot maestro

El robot maestro debe contar con un menú, donde el usuario deberá seleccionar previamente que acción o secuencia de movimientos desea enseñarle al robot discípulo, se analizaran las acciones que se muestran en el menú de la siguiente imagen.

Figura 30. **Aprendizaje por imitación**



Fuente: Documentación descargable en http://software.intel.com/sites/havok/. Consulta: 15 de diciembre de 2011.

5.1.1.1. Enseñar a saludar

Si el usuario selecciona la opción Enseñar a saludar únicamente se ejecutara el aprendizaje mostrando el modelo frente al otro realizando la acción.

Figura 31. **Enseñar a saludar**

Fuente: Documentación descargable en http://software.intel.com/sites/havok/. Consulta: 15 de diciembre de 2011.

Para esta animación es necesario crear un archivo *.hkx (archivo de animación HAVOK) que contenga el ciclo que genera la secuencia de movimientos. Este archivo puede generarse utilizando alguna herramienta de animación como Autodesk 3ds Max, y utilizando la herramienta Havok Content Tools que es un complemento para la exportación de archivos binarios (*.hkx) que pueden ser interpretados por el SDK de Havok. La versión descargable del SDK trae un archivo hkWaveLoop.hkx para la animación del saludo. El código para la carga de este archivo se muestra a continuación.

Tabla III. **Carga de secuenciad e movimientos**

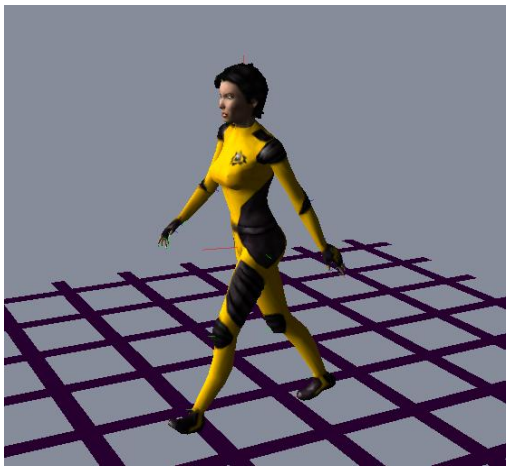
```
//Definimos el path de nuestra animación mediante un buffer
hkStringBuf assetFile("Resources/Animation/HavokGirl/hkWaveLoop.hkx");

//Casteamos la variable assetFile a char*
hkAssetManagementUtil::getFilePath(assetFile);

//Cargamos la animación en un contenedor raiz
hkRootLevelContainer* container = m_loader->load( assetFile.cString() );

//Casteamos de un contenedor raiz a un contenedor e animación
hkaAnimationContainer* ac = reinterpret_cast<hkaAnimationContainer*>(
container->findObjectByType( hkaAnimationContainerClass.getName() ));
```

Fuente: elaboración propia.

5.1.1.2. Enseñar a caminar

Si el usuario selecciona la opción Enseñar a caminar, le aparecerá una opción para definir la ruta a seguir desde una vista de planta, en donde usando una cuadricula podrá definir los vértices y ejecutar el aprendizaje.

Figura 32. **Enseñando a caminar**



Fuente: Documentación descargable en http://software.intel.com/sites/havok/. Consulta: 15 de diciembre de 2011.

La versión descargable del SDK trae un archivo hkWalkLoop.hkx para la animación del saludo. El código para la carga de este archivo se muestra a continuación.

Tabla IV. **Carga de animación para saludar**

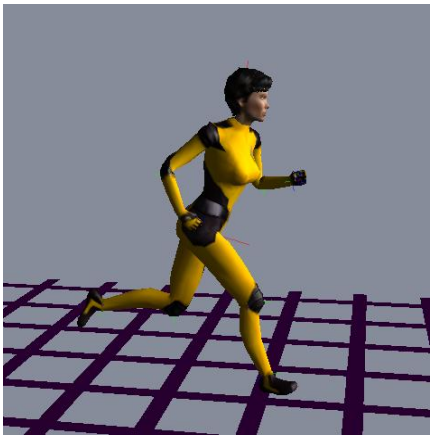
```
//Definimos el path de nuestra animación mediante un buffer
hkStringBuf assetFile("Resources/Animation/HavokGirl/hkWalkLoop.hkx");

//Casteamos la variable assetFile a char*
hkAssetManagementUtil::getFilePath(assetFile);

//Cargamos la animación en un contenedor  aíz
hkRootLevelContainer* container = m_loader->load( assetFile.cString() );

//Casteamos de un contenedor  aíz a un contenedor e animación
hkaAnimationContainer* ac = reinterpret_cast<hkaAnimationContainer*>( container-
>findObjectByType( hkaAnimationContainerClass.getName() ));
```

Fuente: elaboración propia.

5.1.1.3. Enseñar a correr

Si el usuario selecciona la opción Enseñar a correr, le aparecerá una opción para definir la ruta a seguir desde una vista de planta, en donde usando una cuadricula podrá definir los vértices y ejecutar el aprendizaje.

Figura 33. **Enseñar a correr**



Fuente: Documentación descargable en http://software.intel.com/sites/havok/. Consulta: 15 de diciembre de 2011.

La versión descargable del SDK trae un archivo hkRunLoop.hkx para la animación Correr. El código para la carga de este archivo se muestra a continuación.

Tabla V. **Carga de animación para correr**

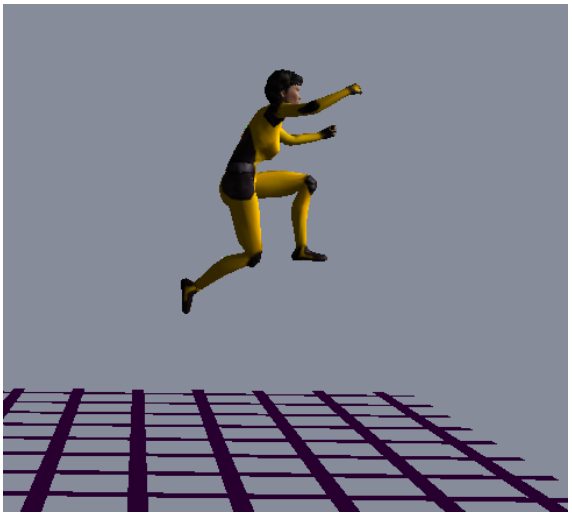
```
//Definimos el path de nuestra animación mediante un buffer
hkStringBuf assetFile("Resources/Animation/HavokGirl/hkRunLoop.hkx");

//Casteamos la variable assetFile a char*
hkAssetManagementUtil::getFilePath(assetFile);

//Cargamos la animación en un contenedor raiz
hkRootLevelContainer* container = m_loader->load( assetFile.cString() );

//Casteamos de un contenedor raiz a un contenedor e animación
hkaAnimationContainer* ac = reinterpret_cast<hkaAnimationContainer*>( container-
>findObjectByType( hkaAnimationContainerClass.getName() ));
```

Fuente: elaboración propia.

5.1.1.4. Enseñar a saltar

Si el usuario selecciona la opción Enseñar a Saltar, le aparecerá una opción para definir el salto que se desea realizar, en donde debe seleccionar el tramo que se desea recorrer con el salgo, para esta opción primero se debe enseñar a correr al robot discípulo.

Figura 34. **Enseñar a saltar**



Fuente: Documentación descargable en http://software.intel.com/sites/havok/. Consulta: 15 de diciembre de 2011.

La versión descargable del SDK trae un archivo hkJumpLandLoop.hkx para la animación para saltar. El código para la carga de este archivo se muestra a continuación.

Tabla VI. **Carga de animación para saltar**

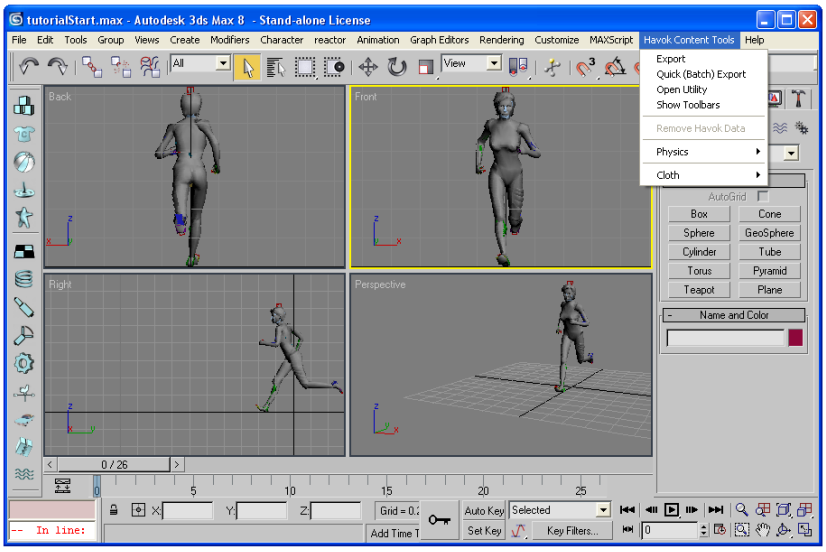
```
//Definimos el path de nuestra animación mediante un buffer
hkStringBuf assetFile("Resources/Animation/HavokGirl/ hkJumpLandLoop.hkx");

//Casteamos la variable assetFile a char*
hkAssetManagementUtil::getFilePath(assetFile);

//Cargamos la animación en un contenedor raiz
hkRootLevelContainer* container = m_loader->load( assetFile.cString() );

//Casteamos de un contenedor raiz a un contenedor e animación
hkaAnimationContainer* ac = reinterpret_cast<hkaAnimationContainer*>( container-
>findObjectByType( hkaAnimationContainerClass.getName() ));
```

Fuente: elaboración propia.

5.1.1.5. Crear nuevas animaciones

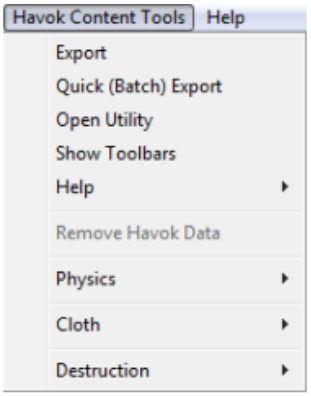
Para crear nuevas animaciones es necesario la utilización de las herramientas Autodesk 3ds Max, y utilizando la herramienta Havok Content Tools desde donde es posible generar los archivos *.hkx.

Figura 35. **Autodesk 3ds mask**



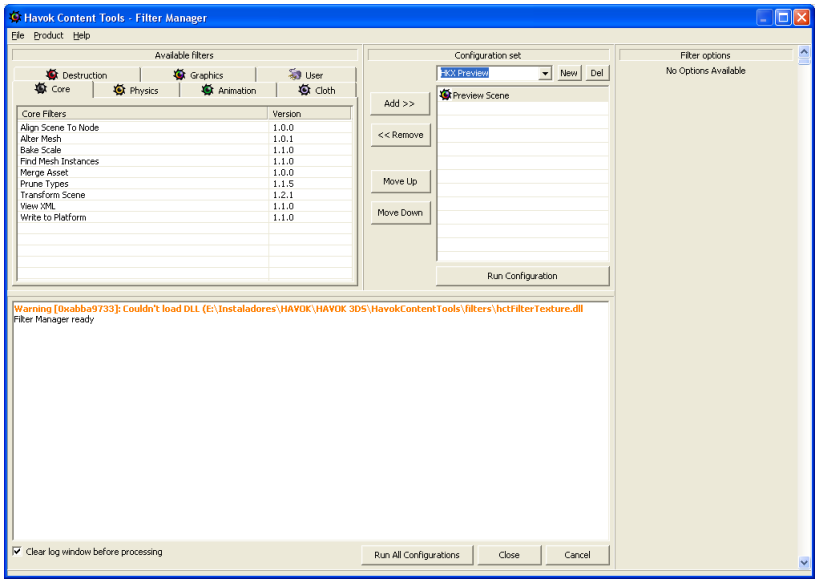
Fuente: Documentación descargable en http://software.intel.com/sites/havok/. Consulta: 15 de diciembre de 2011.

Figura 36. **Menú contextual Autodesk 3ds mask**



Fuente: Documentación descargable en http://software.intel.com/sites/havok/. Consulta: 15 de diciembre de 2011.

Figura 37. **HAVOK content tool**



Fuente: Documentación descargable en http://software.intel.com/sites/havok/. Consulta: 15 de diciembre de 2011.

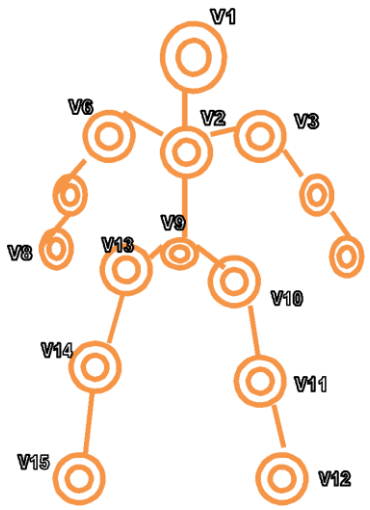
5.2. Variable posición

En esta sección se analizan todos los aspectos necesarios para la recolección de los datos que corresponden a las diversas posiciones que el modelo maestro puede adoptar durante el proceso de aprendizaje.

5.2.1. El grafo tridimensional

Inicialmente para poder definir las coordenadas cartesianas en tres ejes como variables de entrada es necesario enumerar cada vértice del grafo, como se puede apreciar a continuación.

Figura 38. **Figura tridimensional**

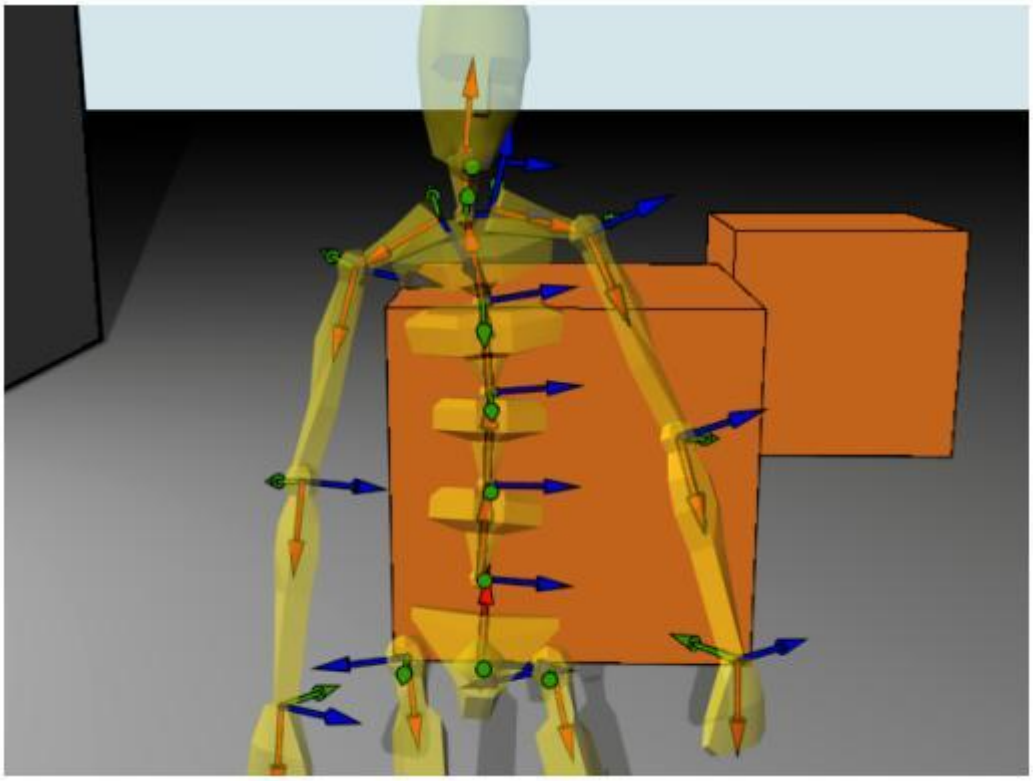


Fuente: elaboración propia.

Después de numerar los vértices se debe ver las coordenadas cartesianas de cada vértice respecto del vértice que le precede, por ejemplo según la siguiente imagen se puede ver que el vértice V3 precede al vértice V4, es decir, que se va a tomar las coordenadas x, y, z del vértice V4 respecto del vértice V3,

en otras palabras, el vértice V4 se vería como un punto en el planocartesiano del vértice V3.

Figura 39. **Modelo tridimensional**



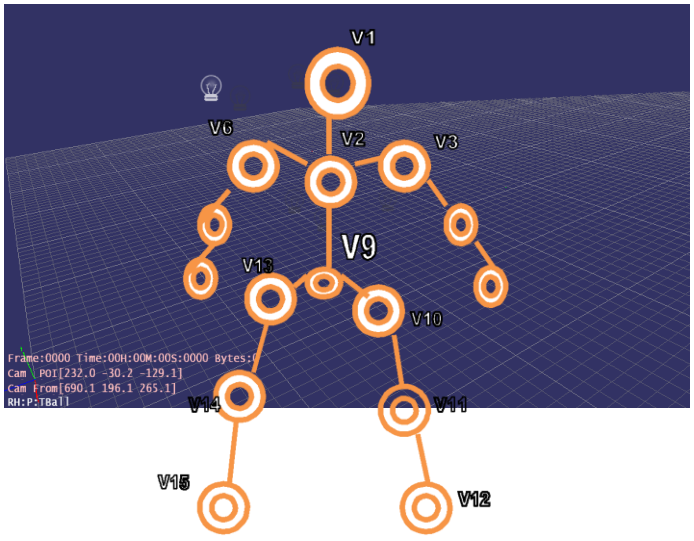
Fuente: Documentación descargable en http://software.intel.com/sites/havok/. Consulta: 15 de diciembre de 2011.

El porqué se toman las coordenadas x, y, z de cada vértice respecto del que le precede se debe a que la red neuronal debe poder generar una secuencia de movimientos aprendida que repercuta en los vértices en forma de cascada. En otras palabras un cambio en la posición de un vértice afectara la posición de todos sus vértices hijos.

Para referirse a aun vértice que precede a otro, en este documento se le llamará vértice de referencia, porque es sobre el plano cartesiano de este vértice que el otro vértice es proyectado.

Por medio de este sistema se enumeraron quince vértices, pero se presenta el problema que debe existir un vértice para el cual sus coordenadas cartesianas sean respecto del mundo virtual para poder posicionar el modelo respecto del entorno, para este propósito el vértice nueve es el ideal pues por ser la pelvis del modelo puede funcionar como vértice de referencia a la mitad superior del cuerpo y a la mitad inferior del cuerpo, al vértice ubicado en el centro del mundo virtual que precederá al vértice nueve, se le llamará vértice de referencia cero. Como se muestra en la siguiente figura.

Figura 40. **Vértice de referencia cero**



Fuente: elaboración propia.

De esta forma, se obtiene un prototipo del formato de entrada para la posición total del modelo a imitar, para la red neuronal, es decir, el grafo representado por medio de la siguiente matriz.

Tabla VII. **Vértices de robot virtual**

Vértice	**Vértice de Referencia**	**X**	**Y**	**Z**
V1	V2	54.35826279	95.236054	53.3851887
V2	V9	22.85689793	30.8521964	90.8232256
V3	V2	8.335508006	90.1606397	14.7333883
V4	V3	8.365976397	11.5340253	15.7803157
V5	V4	82.03450214	64.8241122	64.2154763
V6	V2	37.30181535	50.6462965	50.0843834
V7	V6	62.76749901	25.9664013	81.4933261
V8	V7	9.712110769	61.6875929	62.617009
V9	V0	29.25703303	14.9611387	12.6512228
V10	V9	31.46522487	38.7310019	14.5983226
V11	V10	50.00706045	14.1918603	88.5106598
V12	V12	12.90813689	38.7213429	98.7667775
V13	V9	0.941653737	62.5543672	47.9014826
V14	V13	9.009364846	56.8306951	47.5874306
V15	V14	34.56158605	38.7185153	24.9096831

Fuente: elaboración propia.

Para referirse a esta matriz se le ha llamado matriz de posición.

Pero no se debe dejar de tomar en cuenta que esta matriz va cambiando respecto del tiempo, y el tiempo es una variable de entrada para realizar el aprendizaje, a continuación se describe como construir la secuencia de entrada según los tiempos de realizar cada movimiento.

5.3. Variable posición respecto al tiempo

Todo movimiento toma cierto tiempo en realizarse pero, también se debe considerar que todo movimiento sobre todo humano, consta de varios sub-movimientos, por ejemplo para saludar primero se debe levantar primero el brazo, y luego mover la mano de un lugar a otro, y por ende cada sub-movimiento toma una fracción del tiempo total en realizarse.

Dentro de las actuales herramientas de animación 3D una forma de simular el movimiento al ojo humano es la utilización de frames o fotogramas, un fotograma es una imagen particular que forma parte de una sucesión de imágenes que componen una animación, cada frame tarda cierto tiempo en pantalla antes de mostrar el siguiente, hasta que se completa toda la animación, en cada nuevo frame cambia la posición de los objetos dentro de la animación, el tiempo en que tarda un frame en mostrarse se puede decidir con las herramientas que se estén utilizando, por ejemplo se puede tener una frecuencia de 125 frames por segundo, que por supuesto perdería mucha calidad de video si esta frecuencia fuera reducida a 25 frames por segundo. Esto indica que si se tuviera la matriz de posición del grafo tridimensional que representa al modelo humanoide a imitar, esta matriz estaría cambiando por cada frame que se muestra, en caso de estar en ejecución un

movimiento del modelo humanoide a imitar. Y por ejemplo con una frecuencia de 5 frames por segundo, se obtendría la siguiente tabla:

Tabla VIII. **Matriz de posición a 5 frames por segundo**

Instante de Tiempo	Frame	Matriz de Posición
t = Segundo 1	Frame1	MATRIZ1
t = Segundo 1	Frame2	MATRIZ2
t = Segundo 1	Frame3	MATRIZ3
t = Segundo 1	Frame4	MATRIZ4
t = Segundo 1	Frame5	MATRIZ5
t = Segundo 2	Frame6	MATRIZ6
t = Segundo 2	Frame7	MATRIZ7
t = Segundo 2	Frame8	MATRIZ8
t = Segundo 2	Frame9	MATRIZ9
t = Segundo 2	Frame10	MATRIZ10

Fuente: elaboración propia.

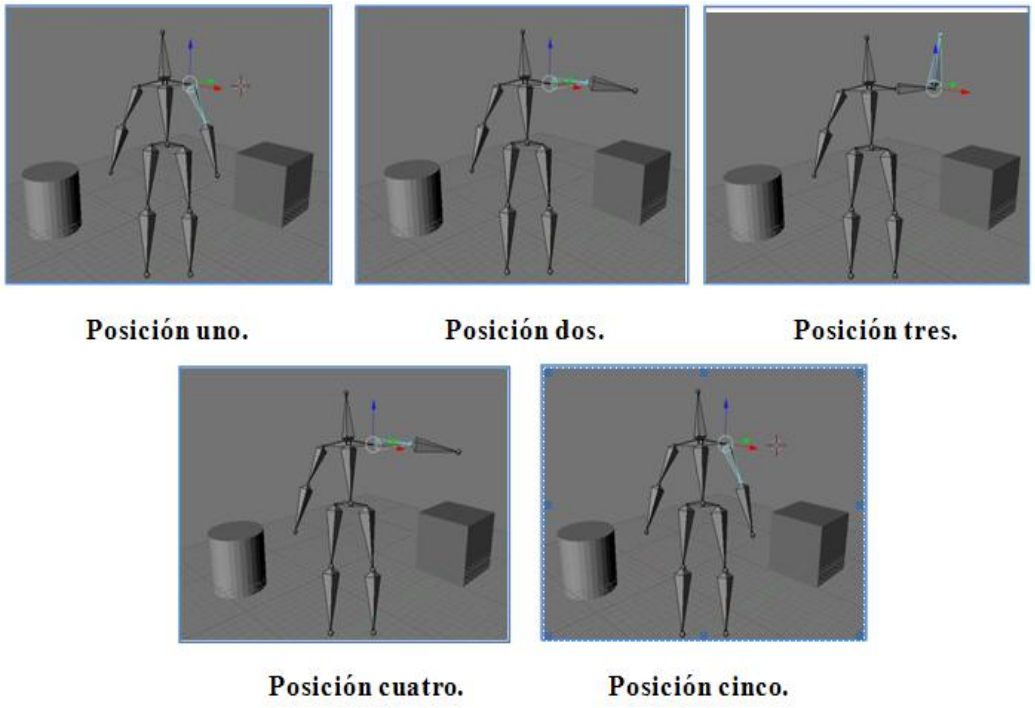
A simple vista es posible notar que de obtener una matriz por cada frame generaría en poco tiempo demasiada información para procesar, que posiblemente la red neuronal no se daría abasto para atender.

Uno de los procesos interesantes del cerebro humano es la abstracción, que trata de sustraer de las cosas más importantes para identificar los objetos y poder agrupar y asociar lo que se ve dependiendo del concepto que se abstrajo de cada cosa. Para poder evitar manejar demasiada información y minimizar la cantidad de matrices que la red neuronal tenga que procesar se debe entonces definir e implementar un proceso de abstracción de la secuencia del movimiento que se detalla a continuación.

5.4. Proceso de abstracción del movimiento

Para poder minimizar la cantidad de matrices de posición a procesar por la red neuronal sin afectar cantidad de información que se necesita para realizar el aprendizaje implica analizar cómo en que frames específicos se ha de obtener la matriz de posición, por ejemplo tómese de ejemplo el movimiento del brazo descrito por las siguientes imágenes

Figura 41. **Proceso de abstracción de movimiento**



Fuente: elaboración propia.

Automáticamente el cerebro autocompleta con estas imágenes todo el movimiento, es decir, por medio de esta muestra de cinco de posiciones del modelo se puede tener la idea de la secuencia total, por ejemplo que de la

posición uno a la posición dos se mueve el brazo hasta llegar a la altura del brazo, pero para llegar a ese punto cada articulación cambia su posición x, y, z respecto de su articulación padre pasando por una cantidad enorme de puntos previos a la posición final resultante de la posición dos.

Si para pasar de una posición a otra tomará un segundo y con una frecuencia de veinte y cinco frames por segundo, se obtendría las siguientes tablas:

Tabla IX. **Matriz de posición a 25 frames por segundo**

Segundo t	Vértice de Referencia	Matriz De Posición		t+1	Vértice de Referencia	Matriz De Posición
t	Frame [(25*t)+1]	Matriz [(25*t)+1]		t+1	Frame [(25*(t+1))+1]	Matriz [(25*(t+1))+1]
t	Frame [(25*t)+2]	Matriz [(25*t)+2]		t+1	Frame [(25*(t+1))+2]	Matriz [(25*(t+1))+2]
t	Frame [(25*t)+3]	Matriz [(25*t)+3]		t+1	Frame [(25*(t+1))+3]	Matriz [(25*(t+1))+3]
t	Frame [(25*t)+4]	Matriz [(25*t)+4]		t+1	Frame [(25*(t+1))+4]	Matriz [(25*(t+1))+4]
t	Frame [(25*t)+5]	Matriz [(25*t)+5]		t+1	Frame [(25*(t+1))+5]	Matriz [(25*(t+1))+5]
t	Frame [(25*t)+6]	Matriz [(25*t)+6]		t+1	Frame [(25*(t+1))+6]	Matriz [(25*(t+1))+6]
t	Frame [(25*t)+7]	Matriz [(25*t)+7]		t+1	Frame [(25*(t+1))+7]	Matriz [(25*(t+1))+7]
t	Frame [(25*t)+8]	Matriz [(25*t)+8]		t+1	Frame [(25*(t+1))+8]	Matriz [(25*(t+1))+8]
t	Frame [(25*t)+9]	Matriz [(25*t)+9]		t+1	Frame [(25*(t+1))+9]	Matriz [(25*(t+1))+9]
t	Frame [(25*t)+10]	Matriz [(25*t)+10]		t+1	Frame [(25*(t+1))+10]	Matriz [(25*(t+1))+10]
t	Frame [(25*t)+11]	Matriz [(25*t)+11]		t+1	Frame [(25*(t+1))+11]	Matriz [(25*(t+1))+11]
t	Frame [(25*t)+12]	Matriz [(25*t)+12]		t+1	Frame [(25*(t+1))+12]	Matriz [(25*(t+1))+12]
t	Frame [(25*t)+13]	Matriz [(25*t)+13]		t+1	Frame [(25*(t+1))+13]	Matriz [(25*(t+1))+13]
t	Frame [(25*t)+14]	Matriz [(25*t)+14]		t+1	Frame [(25*(t+1))+14]	Matriz [(25*(t+1))+14]
t	Frame [(25*t)+15]	Matriz [(25*t)+15]		t+1	Frame [(25*(t+1))+15]	Matriz [(25*(t+1))+15]
t	Frame [(25*t)+16]	Matriz [(25*t)+16]		t+1	Frame [(25*(t+1))+16]	Matriz [(25*(t+1))+16]
t	Frame [(25*t)+17]	Matriz [(25*t)+17]		t+1	Frame [(25*(t+1))+17]	Matriz [(25*(t+1))+17]
t	Frame [(25*t)+18]	Matriz [(25*t)+18]		t+1	Frame [(25*(t+1))+18]	Matriz [(25*(t+1))+18]
t	Frame [(25*t)+19]	Matriz [(25*t)+19]		t+1	Frame [(25*(t+1))+19]	Matriz [(25*(t+1))+19]
t	Frame [(25*t)+20]	Matriz [(25*t)+20]		t+1	Frame [(25*(t+1))+20]	Matriz [(25*(t+1))+20]
t	Frame [(25*t)+21]	Matriz [(25*t)+21]		t+1	Frame [(25*(t+1))+21]	Matriz [(25*(t+1))+21]
t	Frame [(25*t)+22]	Matriz [(25*t)+22]		t+1	Frame [(25*(t+1))+22]	Matriz [(25*(t+1))+22]
t	Frame [(25*t)+23]	Matriz [(25*t)+23]		t+1	Frame [(25*(t+1))+23]	Matriz [(25*(t+1))+23]
t	Frame [(25*t)+24]	Matriz [(25*t)+24]		t+1	Frame [(25*(t+1))+24]	Matriz [(25*(t+1))+24]
t	Frame [(25*t)+25]	Matriz [(25*t)+25]		t+1	Frame [(25*(t+1))+25]	Matriz [(25*(t+1))+25]

Fuente: elaboración propia.

En las tablas anteriores las matrices que realmente serian suficientes para entender el movimiento serian las que están marcadas en rojo, pues son las posiciones resultantes, pero esto aplica para un movimiento sencillo de brazo donde cada posición resultante se da justamente al iniciar otro segundo, es decir, justamente cuando termina un segundo y empieza otro, los cambios en un nodo se detienen y empiezan en otro, por esta razón esa matriz es justamente la que será necesario para abstraer todo el movimiento, a esta matriz en este documento se le llamará matriz de posición resultante.

Pero que pasa en secuencias de movimientos complicadas donde son varias extremidades las que se mueven en paralelo, lo que ocurre es que se tendría muchos nodos cambiando de desplazamiento sobre sus ejes x, y, z, o deteniéndose o dejando de desplazarse en una dirección de un eje para desplazarse en sentido contrario, todo esto simultáneamente respecto de otros nodos, para este caso se han definido tres estados posibles para un nodo sobre sus tres ejes.

La siguiente tabla muestra los posibles estados por los que puede estar un vértice respecto de su eje.

Tabla X. **Estados de vértices respecto a su eje**

Vértice	**X**	**Y**	**Y**
Vi	En Desplazamiento Positivo	En Desplazamiento Positivo	En Desplazamiento Positivo
Vi	En Desplazamiento Negativo	En Desplazamiento Negativo	En Desplazamiento Negativo
Vi	Detenido	Detenido	Detenido

Fuente: elaboración propia.

Por ejemplo si la siguiente tabla corresponde al frame[(frecuencia*t)+i] donde 0<=i<=25, cada nodo en cada vértice tiene uno de los tres estados definidos anteriormente.

Tabla XI. **Frame de frecuencia**

Vértice	Vértice de Referencia	X	Y	Z
V1	V2	En desplazamiento negativo	En desplazamiento positivo	Detenido
V2	V9	En desplazamiento positivo	En desplazamiento negativo	Detenido
V3	V2	Detenido	En desplazamiento negativo	En desplazamiento positivo
V4	V3	En desplazamiento positivo	En desplazamiento positivo	Detenido
V5	V5	En desplazamiento positivo	En desplazamiento negativo	Detenido
V6	V2	Detenido	En desplazamiento negativo	Detenido
V7	V6	En desplazamiento negativo	En desplazamiento positivo	Detenido
V8	V7	En desplazamiento negativo	En desplazamiento negativo	Detenido
V9	V0 (Vértice de referencia 0)	Detenido	En desplazamiento positivo	En desplazamiento negativo
V10	V9	En desplazamiento positivo	En desplazamiento negativo	Detenido
V11	V10	En desplazamiento negativo	Detenido	En desplazamiento positivo
V12	V12	En desplazamiento positivo	En desplazamiento negativo	Detenido
V13	V9	Detenido	En desplazamiento positivo	Detenido
V14	V13	En desplazamiento positivo	En desplazamiento negativo	En desplazamiento positivo
V15	V14	En desplazamiento negativo	En desplazamiento negativo	Detenido

Fuente: elaboración propia.

Cada vértice de la tabla anterior se mantiene en un estado por cada eje por un número determinado de frames pero en cierto momento este vértice cambia su estado respecto de alguno de sus ejes, tomando el primer vértice de ejemplo:

Tabla XII. **Vértice V1 en tiempo t**

Vértice	Vértice de Referencia	X	Y	Z
V1	V2	En Desplazamiento Negativo	En Desplazamiento positivo	Detenido

Fuente: elaboración propia.

Pero en algún momento del tiempo el vértice cambia su estado en uno de sus ejes o en más de uno.

Tabla XIII. **Vértice v1 en tiempo t+n**

Vértice	Vértice de Referencia	X	Y	Z
V1	V2	Detenido	En Desplazamiento positivo	Detenido

Fuente: elaboración propia.

De esta manera, mientras todos los vértices se mantengan en el mismo estado para cada uno de sus ejes, ningún vértice ha llegado a un punto resultante, por lo que mientras no haya un cambio en algún estado en algún eje

en algún vértice, no hay matriz resultante, esto indica que el momento donde se debe almacenar una matriz de posición como matriz de posición resultante, es cuando algún vértice cambie su estado en alguno de sus ejes. Con esta política, con una frecuencia de veinte y cinco frames por segundo, se obtendría la siguiente tabla donde las matrices resultantes están marcadas en rojo.

Tabla XIV. **Frames por segundo en tiempo**

Segundo t	Vértice de Referencia	Matriz De Posición		t+1	Vértice de Referencia	Matriz De Posición
t	Frame [(25*t)+1]	Matriz [(25*t)+1]		t+1	Frame [(25*(t+1))+1]	Matriz [(25*(t+1))+1]
t	Frame [(25*t)+2]	Matriz [(25*t)+2]		t+1	Frame [(25*(t+1))+2]	Matriz [(25*(t+1))+2]
t	Frame [(25*t)+3]	Matriz [(25*t)+3]		t+1	Frame [(25*(t+1))+3]	Matriz [(25*(t+1))+3]
t	Frame [(25*t)+4]	Matriz [(25*t)+4]		t+1	Frame [(25*(t+1))+4]	Matriz [(25*(t+1))+4]
t	Frame [(25*t)+5]	Matriz [(25*t)+5]		t+1	Frame [(25*(t+1))+5]	Matriz [(25*(t+1))+5]
t	Frame [(25*t)+6]	Matriz [(25*t)+6]		t+1	Frame [(25*(t+1))+6]	Matriz [(25*(t+1))+6]
t	Frame [(25*t)+7]	Matriz [(25*t)+7]		t+1	Frame [(25*(t+1))+7]	Matriz [(25*(t+1))+7]
t	Frame [(25*t)+8]	Matriz [(25*t)+8]		t+1	Frame [(25*(t+1))+8]	Matriz [(25*(t+1))+8]
t	Frame [(25*t)+9]	Matriz [(25*t)+9]		t+1	Frame [(25*(t+1))+9]	Matriz [(25*(t+1))+9]
t	Frame [(25*t)+10]	Matriz [(25*t)+10]		t+1	Frame [(25*(t+1))+10]	Matriz [(25*(t+1))+10]
t	Frame [(25*t)+11]	Matriz [(25*t)+11]		t+1	Frame [(25*(t+1))+11]	Matriz [(25*(t+1))+11]
t	Frame [(25*t)+12]	Matriz [(25*t)+12]		t+1	Frame [(25*(t+1))+12]	Matriz [(25*(t+1))+12]
t	Frame [(25*t)+13]	Matriz [(25*t)+13]		t+1	Frame [(25*(t+1))+13]	Matriz [(25*(t+1))+13]
t	Frame [(25*t)+14]	Matriz [(25*t)+14]		t+1	Frame [(25*(t+1))+14]	Matriz [(25*(t+1))+14]
t	Frame [(25*t)+15]	Matriz [(25*t)+15]		t+1	Frame [(25*(t+1))+15]	Matriz [(25*(t+1))+15]
t	Frame [(25*t)+16]	Matriz [(25*t)+16]		t+1	Frame [(25*(t+1))+16]	Matriz [(25*(t+1))+16]
t	Frame [(25*t)+17]	Matriz [(25*t)+17]		t+1	Frame [(25*(t+1))+17]	Matriz [(25*(t+1))+17]
t	Frame [(25*t)+18]	Matriz [(25*t)+18]		t+1	Frame [(25*(t+1))+18]	Matriz [(25*(t+1))+18]
t	Frame [(25*t)+19]	Matriz [(25*t)+19]		t+1	Frame [(25*(t+1))+19]	Matriz [(25*(t+1))+19]
t	Frame [(25*t)+20]	Matriz [(25*t)+20]		t+1	Frame [(25*(t+1))+20]	Matriz [(25*(t+1))+20]
t	Frame [(25*t)+21]	Matriz [(25*t)+21]		t+1	Frame [(25*(t+1))+21]	Matriz [(25*(t+1))+21]
t	Frame [(25*t)+22]	Matriz [(25*t)+22]		t+1	Frame [(25*(t+1))+22]	Matriz [(25*(t+1))+22]
t	Frame [(25*t)+23]	Matriz [(25*t)+23]		t+1	Frame [(25*(t+1))+23]	Matriz [(25*(t+1))+23]
t	Frame [(25*t)+24]	Matriz [(25*t)+24]		t+1	Frame [(25*(t+1))+24]	Matriz [(25*(t+1))+24]
t	Frame [(25*t)+25]	Matriz [(25*t)+25]		t+1	Frame [(25*(t+1))+25]	Matriz [(25*(t+1))+25]

Fuente: elaboración propia.

Es posible observar que al momento de ver al personaje moverse, se obtendrán matrices resultantes en diferentes momentos del tiempo, por lo que la matriz resultante M(i+1) es la matriz a la que se debe llegar a partir de la matriz resultante Mi. Por lo que se tiene un conjunto de matrices resultantes como por ejemplo el que se describe a continuación.

Tabla XV. **Ejemplo matriz resultante**

Vértice de referencia	**Matriz de posición resultante**
Frame0	Matriz1
Frame5	Matriz2
Frame16	Matriz3
Frame29	Matriz4
Frame30	Matriz5
Frame31	Matriz6
Frame40	Matriz7
Frame43	Matriz8
Frame45	Matriz9
Frame46	Matriz10

Fuente: elaboración propia.

La red neuronal debe poder imitar esta secuencia tomando en cuenta cuanto el tiempo se tarda en completar un sub-movimiento, la forma de cómo se gestiona esta información se describe a continuación.

5.5. Variable tiempo

Cuando una niña está saltando la cuerda se puede ver que sus saltos son pequeños y de corta duración, pero cuando se ve a un atleta entrenado el salto de longitud que es una prueba consistente en recorrer la máxima distancia posible en el plano horizontal a partir de un salto tras una carrera, se puede ver que aunque la niña también salta, el movimiento es diferente para las extremidades, pero muy importante es notar que el tiempo que tarda el salto de longitud es mucho más largo que el tiempo que tarda la niña cada vez en saltar la cuerda, por lo que para copiar una secuencia de movimientos es necesario que la red neuronal maneje también información de los tiempos que tarda el modelo a imitar en realizar cada sub-movimiento, es decir, lo que tarda en pasar de una matriz resultante a otra matriz resultante. Este tiempo no se medirá en segundos sino en frames. Por ejemplo esto se puede ver en la siguiente tabla:

Tabla XVI. **Matriz resultante**

Vértice de referencia	Matriz de posición resultante	Diferencia de Frames
Frame0	MATRIZ1	0
Frame5	MATRIZ2	5
Frame16	MATRIZ3	11

Fuente: elaboración propia.

De la tabla anterior se puede ver que al modelo le tomó por ejemplo once frames en pasar de la matriz dos a la matriz tres, con esto se puede ver la cantidad de frames como una unidad de tiempo para poder imitar el movimiento, pero la cantidad de frames solo se puede ver como unidad de

tiempo vista desde el punto de vista de la frecuencia, es decir, que con una frecuencia de veinticinco frames por segundo, once frames representarían 0.44 segundos, pero durante la simulación, la cantidad de frames no será tan pequeña, tendrá valores mucho más largos entre cada movimiento.

5.6. Matriz de secuencia

En este punto ya es posible definir la matriz de variables de entrada, que consiste en almacenar en la primera columna las matrices resultantes de coordenadas x, y, z y en la segunda columna la cantidad de frames que se tardó en llegar de la matriz resultante i-1 a la matriz resultante i.

Tabla XVII. **Matriz de secuencia**

Matriz de posición resultante	**Frames que tardo el modelo en llegar a la matriz i**
Matriz1	0
Matriz2	75
Matriz3	45
Matriz4	69
Matriz5	50
Matriz6	65
Matriz7	75
Matriz8	68
Matriz9	46
Matriz10	35

Fuente: elaboración propia.

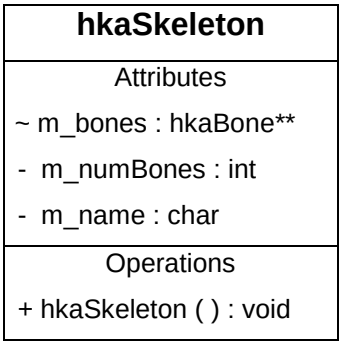
5.7. Clases y métodos a utilizar

A continuación se presenta una descripción práctica de todos los puntos relevantes de la Biblioteca Base Havok para la presente investigación. También se muestra cómo el comportamiento por defecto puede ser extendido o reemplazado por muchas de estas funciones del sistema de animación mencionado.

5.7.1. Clase hkaSkeleton

Esta clase contiene variedad de métodos estáticos para la manipulación de posiciones sobre un esqueleto dentro de una animación hecha en Havok. Esta clase contiene la jerarquía que define la relación padre / hijo entre los huesos.

Figura 42. **Descripción de clase hkaSkeleton**



Fuente: elaboración propia.

Tabla XVIII. **Descripción de atributos y métodos**

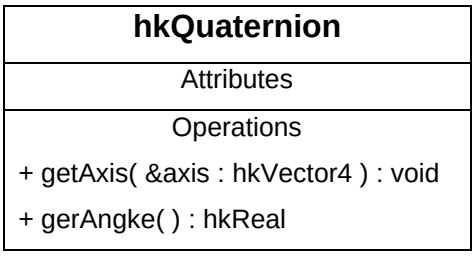
Atributo	Descripción
m_bones	Hace referencia al conjunto de huesos que comprende el esqueleto entero.
m_numBones	Almacena el número de huesos que contiene el esqueleto
m_name	Almacena el nombre del esqueleto
Método	**Descripción**
hkaSkeleton()	Constructor

Fuente: elaboración propia.

5.7.2. Clase hkQuaternion

Almacena las propiedades principales para la realización de rotaciones en tres dimensiones. Estas funciones son especialmente importantes pues están directamente relacionadas con el movimiento de un modelo articulado.

Figura 43. **Clase hkQuaternion**



Fuente: elaboración propia.

Tabla XIX. **Métodos clase hkCuaternion**

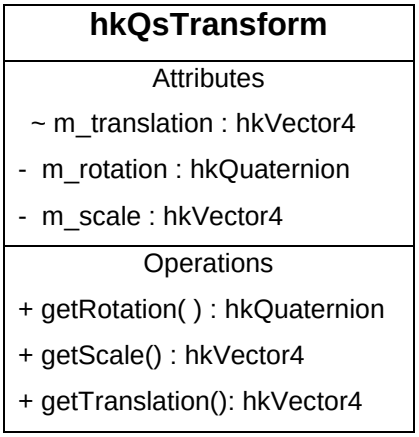
Método	Descripción
getAxis ()	Recibe por referencia el vector axis para almacenar en el las coordenadas x, y, z del objeto al cual pertenece en este caso del hueso.

Fuente: elaboración propia.

5.7.3 Clase hkQsTransform

Descripción: esta clase como tal es una representación en tres aspectos relevantes a un objeto dentro de una animación, estos aspectos son transformación, rotación y escala.

Figura 44. **Diseño clase hkQsTransform**



Fuente: elaboración propia.

Tabla XX. **Atributos clase hkQsTransform**

Atributo	Descripción
m_translation	Variable que hace referencia al componente de traslación mediante un vector.
m_rotation	Variable que hace referencia al componente de rotación representado mediante la clase hkQuaternion.
m_scale	Variable que hace referencia al componente de escala mediante un vector.

Fuente: elaboración propia.

Tabla XXI. **Métodos clase hkQsTransform**

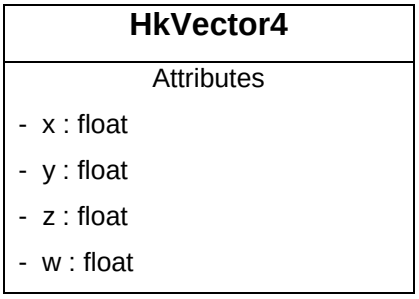
Método	Descripción
getRotation()	Retorna un objeto de tipo hkQuaternion para poder tener acceso al componente de rotación.
getTranslation()	Retorna un vector para poder tener acceso al componente de traslación.
getScale()	Retorna un vector para poder tener acceso al componente de escala.

Fuente: elaboración propia.

5.7.4. Clase hkVector4

Descripción: esta clase proporciona los métodos necesarios para representar cualquier punto en un espacio tridimensional, utilizando cuatro valores de tipo flotante, x, y, z, w, donde w se define con el fin de almacenar un valor asociado al punto x, y, z

Figura 45. **Diseño clase hkVector4**

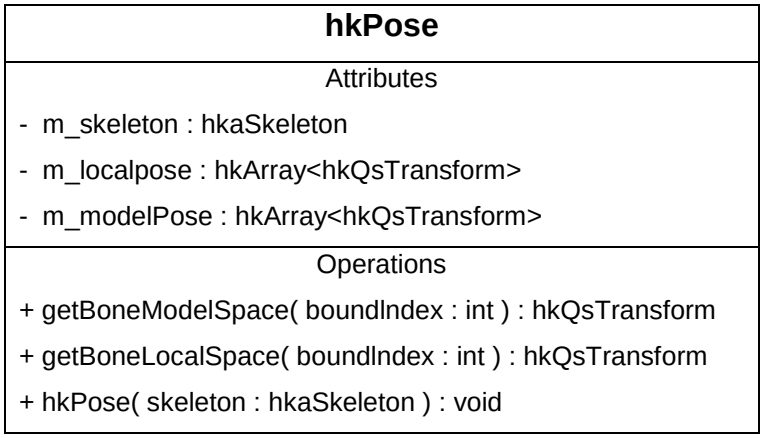


Fuente: elaboración propia.

5.7.5. Clase hkPose

Descripción: Esta clase provee los métodos para poder almacenar y manipular posiciones sobre los esqueletos utilizados dentro de una animación hecha con Havok, esta clase posee un constructor que recibe un esqueleto como parámetro sobre el cual se asocia la instancia.

Figura 46. **Diseño clase hkPose**



Fuente: elaboración propia.

Tabla XXII. **Atributos clase hkpose**

Atributo	Descripción
m,_skeleton	Es el esqueleto sobre el cual se hace referencia.
m_localPose	Vector de objetos hkQsTransform para hacer referencia la posición del hueso respecto de su padre.
m_modelPose	Vector de objetos hkQsTransform para hacer referencia la posición del hueso respecto del esqueleto.

Fuente: elaboración propia.

Tabla XXIII. **Métodos clase hkpose**

Método	Descripción
getBoneModelSpace(boundIndex: int)	Retorna el hkQsTransform necesario para modificar la posición del hueso con índice bondIndex sobre el esqueleto.
getBoneLocalSpace(boundIndex: int)	Retorna el hkQsTransform necesario para modificar la posición del hueso con índice bondIndex respecto de su hueso padre.
hkPose(skeleton: hkaSkeleton)	Crea una instancia nueva de hkPose con el esqueleto skeleton como parámetro, para hacer referencia sobre él.

Fuente: elaboración propia.

Con las clases descritas anteriormente, dentro de una animación es posible obtener la posición de un hueso relativa a su hueso padre utilizando el siguiente código.

```
hkaPose pose (m_skeleton);
hkVector4 posicion;
pose.getBoneLocalSpace(1/*número de hueso*/).getRotation()
.getAxis(posicion/*pasa por referencia*/);
x = posicion(0); y = posicion(1); z = posicion(2);
```

En el código anterior se puede observar que m_skeleton sería el esqueleto, pose el objeto que proporcional hkQsTransform para un hueso especifico, y almacenando los valores de las coordenadas en x, y, z en el vector posición por medio de los métodos proporcionados por hkQsTransform.

5.7.6. Clase hkPWorld

Descripción: Havok proporciona una clase llamada hkpWorld que contiene todos los métodos y atributos para representar el mundo virtual, esta clase actua como contenedor para los objetos físicos en la simulación, y también se utiliza para avanzar cada paso de la simulación en el tiempo.

Figura 47. **Diseño clase hkpWorld**

hkpWorld
Attributes
Operations + stepDeltaTime (DeltaTime : float) : void

Fuente: elaboración propia.

A continuación se describe que método está implicado en la obtención del número de frame sobre el cual se encuentre la animación.

- stepDeltaTime(DeltaTime): lo que hace este método permite que la animación avance, es decir debe ser llamado dentro de un ciclo, para que la animación pueda darse, y se actualice el video que se le presenta al usuario.

- Parámetro DeltaTime: este parámetro especifica el ritmo de tiempo de la animación, es decir que tan rápido va a avanzar la animación, este tiempo debe especificársele en referencia a la frecuencia, por ejemplo si la frecuencia que se desea es de sesenta frames por segundo, deltaTime debe ser inicializado con el valor 1/60 = 0.015.

Entonces siempre que esta función sea llamada hay que actualizar la variable global que almacenará el frame actual, utilizando esta información ya es posible construir una matriz de secuencia para la red neuronal como la descrita anteriormente.

Tabla XXIV. **Matriz de secuencia**

Matriz de posición resultante	**Frames que tardo el modelo en llegar a la matriz i**
Matriz1	0
Matriz2	75
Matriz3	45
Matriz4	69
Matriz5	50
Matriz6	65
Matriz7	75
Matriz8	68
Matriz9	46
Matriz10	35

Fuente: elaboración propia.

6. ANÁLISIS Y DISEÑO DE LA RED NEURONAL ESPECULAR

Una vez que ya se tiene toda la información que se desea sea procesada como variables de entrada ya es posible buscar un modelo de red neuronal que se adapte y consiga los resultados deseados. El propósito de este capítulo es encontrar el diseño adecuado de una red neuronal artificial basada en un modelo perceptrón multicapa, que consiga muy acertadamente lograr imitar una secuencia de movimientos dentro de un entorno tridimensional, la que para este documento es llamada red neuronal especular.

6.1. El modelo de Rossenblatt como modelo base

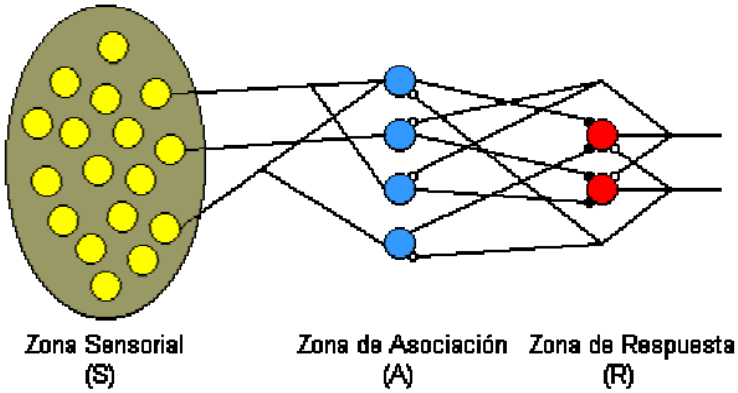
Para poder realizar el diseño de la red neuronal primero se tiene que definir que se desea que la red neuronal proporcione como información de salida, a partir de las variables de entrada, que es en este caso la matriz de secuencia definida en el capítulo anterior

Lo que se desea de manera puntual, es que una red neuronal de tipo perceptrón multicapa, a partir de una matriz de secuencia, genere como salidas, una secuencia de movimientos, que al aplicarla a un modelo humanoide, este imite el comportamiento de otro modelo humanoide del cual se obtuvo la matriz de secuencia.

El modelo perceptrón resulta ser el más adecuado como modelo base para el diseño de la red neuronal especular, no solo porque es un modelo de aprendizaje supervisado, sino porque fue creado inicialmente para ilustrar

propiedades fundamentales de sistemas inteligentes en general, inventado por el psicólogo Frank Rosenblatt en el año 1957. El primer modelo de perceptrón fue desarrollado en un ambiente biológico, imitando el funcionamiento del ojo humano, a este modelo se le llamó Modelo del fotoperceptrón de RosenBlatt mostrado en la siguiente imagen.

Figura 48. **Modelo de red neuronal especular**



Fuente: http://perso.wanadoo.es/alimanya/funcion.htm. Consulta: 15 de diciembre de 2011.

De la imagen anterior se puede ver que la zona sensorial estaría formada por la matriz de secuencia, sin embargo en este punto haría falta diseñar la zona de asociación y zona de respuesta.

6.2. La zona de asociación y la zona de respuesta

Para poder diseñar la zona de asociación es necesario primero ver de manera general que información se espera manejar en la zona de respuesta y como se va a proporcionar esta información, para esto es necesario conocer las herramientas con las que se cuenta para el desarrollo del software.

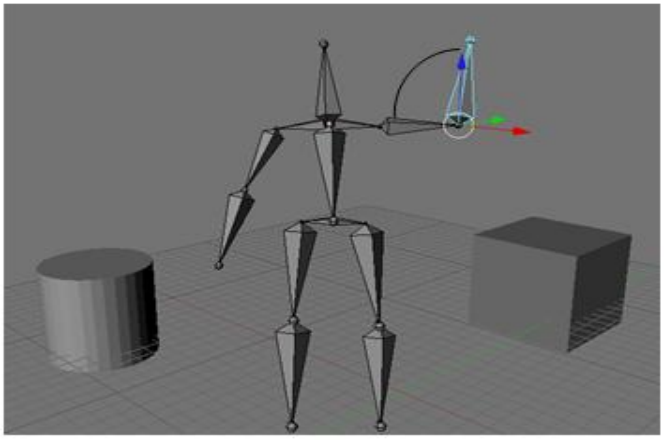
El análisis de este documento se enfoca en utilizar las librerías proporcionadas por el SDK del motor físico Havok, desarrollado por la compañía irlandesa con el mismo nombre, dichas librerías proporcionan clases y métodos para poder manipular esqueletos dentro de una animación tridimensional.

Tal y como se puede ver en el capítulo anterior, la clase hkQsTransform que pertenece a las librerías de Havok, posee atributos y métodos para la realización de rotaciones, traslaciones, y cambios de escala sobre un objeto especifico, de esta clase los métodos más apropiados para obtener la información de la zona de respuesta, son los asociados a los movimientos de rotación.

El objetivo es que la zona de respuesta posea información respecto de la secuencia de movimientos que el modelo que imita debe seguir, pero si esta información se maneja también como una matriz de vértices, como las matrices resultantes definidas en el capítulo anterior, es posible obtener coordenadas x,y,z que no concuerden con lo que el esqueleto pueda lograr respecto de las distancias de sus huesos, es decir existiría un margen de error de las posiciones de los vértices muy difícil de controlar, que solo retrasaría el proceso de aprendizaje, este margen de error queda totalmente eliminado si la secuencia resultante se maneja en términos de rotaciones entre los huesos respecto del vértice padre, utilizando los métodos proporcionados por hkQsTransform.

Figura 49. **hkQsTransfor y esqueleto**

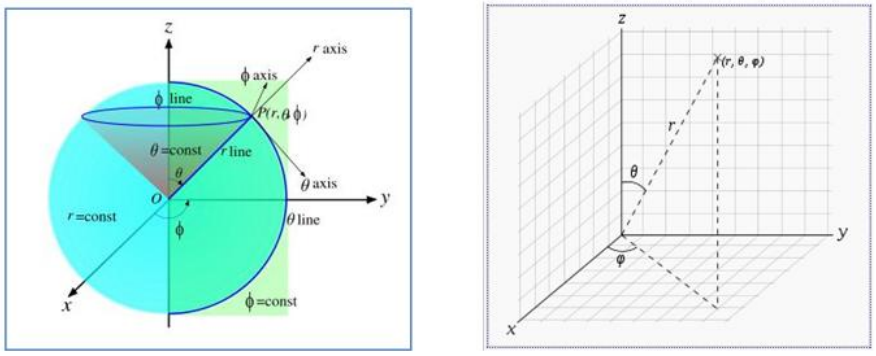
HkQsTransform
Attibutes
- m_translation : hkVector4
- m_rotation : hkQuaternion
- m_scale : hkVector4
Operations
+ getRotation( ) : hkQuaternion
+ getScale( ) : hkVector4
+ getTranslation( ) : hkVector4



Fuente: elaboración propia.

Para poder manejar información de la posición de los vértices de un modelo tridimensional utilizando movimientos de rotación, es necesario ver esta información en términos de ángulos en dicho espacio tridimensional, lo que es posible mediante la utilización de coordenadas esféricas.

Figura 50. **Información en términos de ángulos**

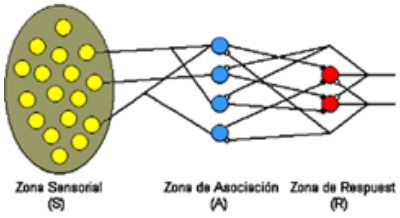


Fuente: http://es.wikipedia.org/wiki/Coordenadas_esf%C3%A9ricas. Consulta: 15 de diciembre de 2011.

- P (Radio): es la distancia entre el punto P y el origen.
- φ (azimuth o longitud) de 0º a 180º es el ángulo entre el eje z y la línea que une el origen y el punto P.
- θ (colatitud o ángulo polar) de 0º a 360º es el ángulo entre el eje X positivo y la línea que une el origen con la proyección del punto P en el plano XY.

Mediante el anterior análisis puede verse a grandes rasgos que lo que se desea, es que de una matriz de secuencia, construida a partir de matrices de posiciones cartesianas resultantes, se obtenga otra matriz de secuencia construida a partir de matrices de posición esféricas resultantes, como se puede ver en la siguiente imagen.

Figura 51. **Matrices de posición esférica resultante**

Matriz de posición esféricas resultante	**Frames que tardo el modelo en llegar a la matriz i**		**Matriz de posición esféricas resultante**	**Frames que tardo el modelo en llegar a la matriz i**
Matriz1	0		Matriz_E1	0
Matriz2	75		Matriz_E2	75
Matriz3	45		Matriz_E3	45
Matriz4	69		Matriz_E4	69
Matriz5	50		Matriz_E5	50
Matriz6	65		Matriz_E6	65
Matriz7	75		Matriz_E7	75
Matriz8	68		Matriz_E8	68
Matriz9	46		Matriz_E9	46
Matriz10	35		Matriz_E10	35

Fuente: elaboración propia.

Es decir que para cada conjunto de coordenadas x, y, z, de cada vértice respecto de su padre dentro de cada matriz de posición resultante, se obtenga mediante el entrenamiento de la red neuronal el conjunto de coordenadas esféricas correspondientes a dicho punto x, y, z.

Tabla XXV. **Matriz de coordenadas x, y, z a esféricas resultantes**

Vértice	Vértice de Referencia	X	Y	Z
V1	V2	54.35826279	95.236054	53.3851887
V2	V9	22.85689793	30.8521964	90.8232256
V3	V2	8.335508006	90.1606397	14.7333883
V4	V3	8.365976397	11.5340253	15.7803157
V5	V4	82.03450214	64.8241122	64.2154763
V6	V2	37.30181535	50.6462965	50.0843834
V7	V6	62.76749901	25.9664013	81.4933261
V8	V7	9.712110769	61.6875929	62.617009
V9	V0	29.25703303	14.9611387	12.6512228
V10	V9	31.46522487	38.7310019	14.5983226
V11	V10	50.00706045	14.1918603	88.5106598
V12	V12	12.90813689	38.7213429	98.7667775
V13	V9	0.941653737	62.5543672	47.9014826
V14	V13	9.009364846	56.8306951	47.5874306
V15	V14	34.56158605	38.7185153	24.9096831

Continuación de la tabla XXV.

Vértice	Vértice de Referencia	P	φ	θ
V1	V2	82.2265527	72.117673	1.52128494
V2	V9	88.8768535	11.9901187	76.391481
V3	V2	64.0584858	14.534059	6.13020632
V4	V3	95.4240084	84.4508569	76.7294508
V5	V4	61.4591925	76.4087356	80.8671965
V6	V2	66.9284851	49.7562656	18.9672046
V7	V6	12.054207	83.6690366	1.2169441
V8	V7	33.3526565	31.4970054	5.19047143
V9	V0	11.6453975	6.664019	64.5104983
V10	V9	33.9576022	25.8578595	69.8934309
V11	V10	4.97952485	88.6104372	78.935601
V12	V12	56.0138375	92.6306063	9.36878117
V13	V9	4.73726167	52.8835558	37.6790167
V14	V13	13.1884599	56.9626799	36.1298085
V15	V14	67.9857597	4.86011435	65.4787555

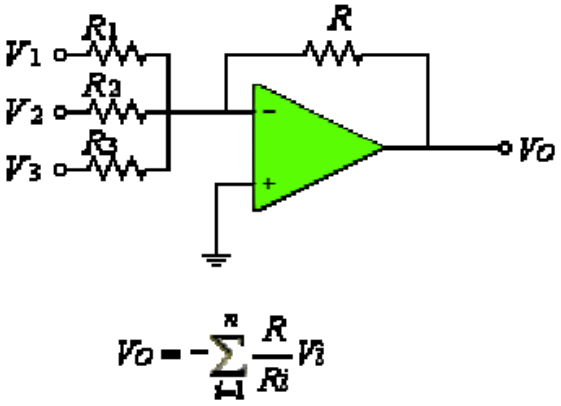
Fuente: elaboración propia.

Esta transformación de coordenadas cartesianas a esféricas seria parte fundamental de las funciones de activación de las neuronas de la red como se describe a continuación.

6.3. Análisis perceptrón multicapa

Un modelo de red neuronal artificial busca en si mismo representar e imitar el proceso de una red neuronal biológica, para entender cómo funciona cada neurona dentro de la red es posible verla en analogía con un amplificador operacional como puede verse en la siguiente imagen.

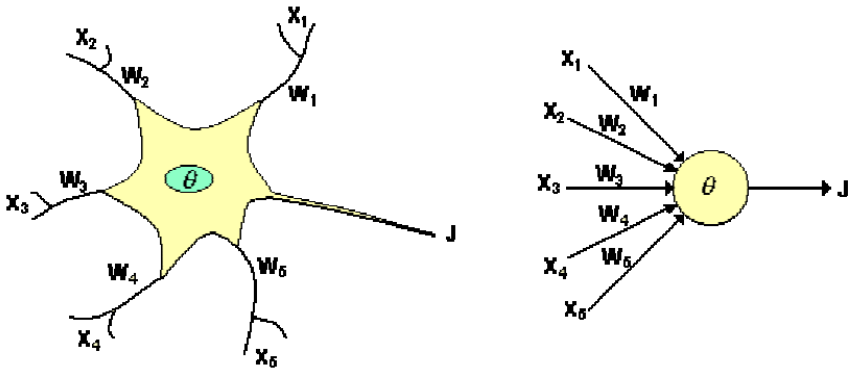
Figura 52. **Perceptrón multicapa**



Fuente: http://sav.us.es. Consulta: 15 de diciembre de 2011.

Se puede observar de la imagen anterior, que el voltaje resultante es igual a la sumatoria de los voltajes entrantes dividido su resistencia, multiplicado por la resistencia sobre el voltaje ponderado, de manera similar una neurona biológica genera una salida a partir de sus múltiples entradas.

Figura 53. **Neurona biológica**

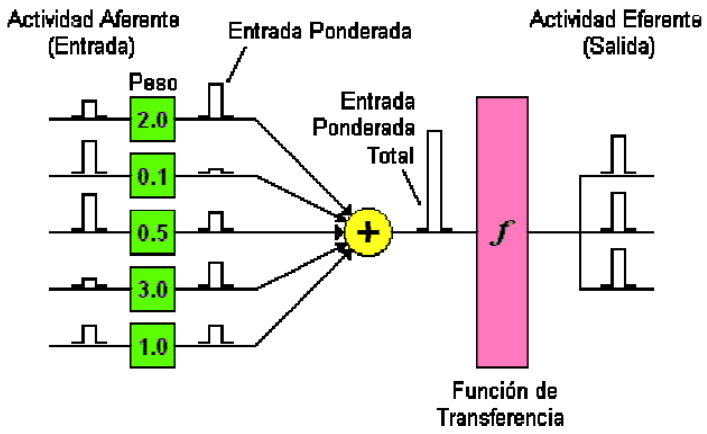


Fuente: http://sav.us.es. Consulta: 15 de diciembre de 2011.

- Donde las entradas Xi son las señales provenientes de otras neuronas y que son capturadas por las dendritas.
- Los pesos Wi son la intensidad de la sinapsis que conecta una neurona con otra.
- Y θ es la función umbral que la neurona debe sobrepasar para activarse, o también llamada función de transferencia. (Dentro de la zona de respuesta, el valor θ está asociado al ángulo de colatitud).

Este proceso ocurre biológicamente dentro de la célula. Cada conexión a una neurona transfiere impulsos eléctricos de información con cierta intensidad a otras neuronas, a esa intensidad se le representa mediante W, cada conexión tiene su propio peso y este peso afecta en si la entrada de información a la neurona ver figura 54.

Figura 54. **Ejemplo de actividad perceptrón**



Fuente: http://sav.us.es. Consulta: 15 de diciembre de 2011.

Figura 55. **Entrada ponderada total**

$$neta_i = \sum_{i=1}^{n} W_i X_i = \vec{X}\,\vec{W}$$

Fuente: elaboración propia.

En la imagen anterior puede verse como las entradas de las neuronas son multiplicadas cada una por un peso que representa la intensidad de la sinapsis, luego estas entradas ya multiplicadas por sus pesos son sumadas para obtener una entrada ponderada total que es evaluada por la función de transferencia, que es quien finalmente proporciona la salida resultante que es pasada a otras neuronas.

Para poder representar cada aspecto de una neurona se utilizará la siguiente notación:

- Pi: entrada a la neurona

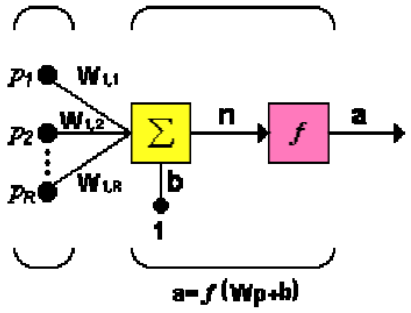
- Wi,j: peso sináptico para la conexión entre la neurona i, y la neurona j de la capa anterior

- b: es una ganancia o pérdida aplicada a la sumatoria ponderada, este valor es común en redes neuronales tipo perceptrón para obtener un aprendizaje más acelerado, su valor inicial es aleatorio igual que los pesos sinapticos.

- n: es la suma ponderada total

- a: es la salida de la neurona que está en función de n.

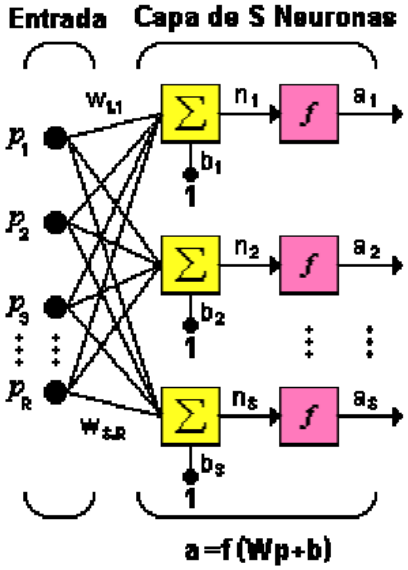
Figura 56. **Representación perceptron función**



Fuente: http://sav.us.es. Consulta: 15 de diciembre de 2011.

Al momento de tomar este modelo de neurona, y utilizarlo para formar una capa en la red neuronal se obtendría la siguiente estructura:

Figura 57. **Representación perceptrón multicapa**



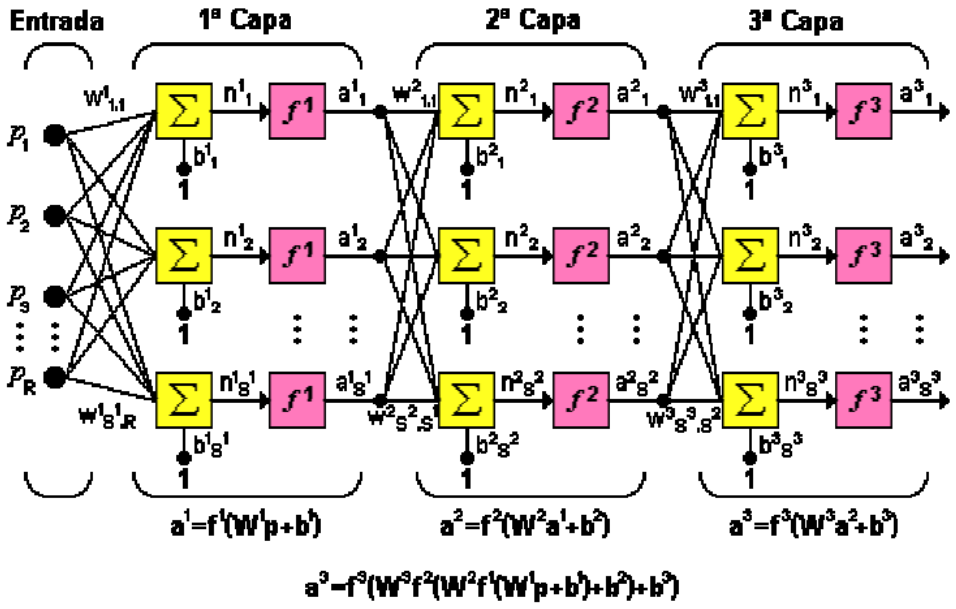
Fuente: http://sav.us.es. Consulta: 15 de diciembre de 2011.

Para la imagen anterior, puede verse que cada entrada se dirige a cada neurona de la primera capa de la red neuronal, pero cada conexión con su peso sináptico diferente, sin embargo el modelo perceptrón puede tener dos diferentes diseños:

- Totalmente conectado: cada salida de una neurona de la capa i es entrada de todas las neuronas de la capa i+1.
- Localmente conectado: cada neurona de la capa i es entrada de una serie de neuronas (región) de la capa i+1, es decir que no todas las neuronas se conectan con todas las neuronas de la capa siguiente.

Por ejemplo para un modelo totalmente conectado se tendría el siguiente diseño:

Figura 58. **Perceptron completamente acoplado**



Fuente: http://sav.us.es. Consulta: 15 de diciembre de 2011.

De la imagen anterior se puede ver que cada capa tiene varias salidas que son inyectadas a la siguiente capa en cada neurona con pesos distintos nuevamente, y cada capa tiene una función de activación basada en las salidas de su capa anterior.

Para saber si para diseñar la red neuronal es necesario un modelo totalmente conectado o localmente conectado es necesario primero definir por neurona y capa que se desea querealicen dependiendo de las entradas que se reciban.

6.4. Análisis y diseño de la red neuronal

Como parte de cualquier estructura organizacional, es común ver que en dicha estructura, todos los elementos tienen funciones diferentes y cada quien se dedica y especializa en lograr sus tareas. De la misma manera el cerebro humano posee diversos tipos de neuronas, donde cada una está asociada a procesos específicos del cuerpo, por esto se definen tres tipos de neuronas para el diseño de la red neuronal especular.

- Neuronas de estimación de movimiento: son las encargadas de traducir de coordenadas cartesianas a coordenadas esféricas.
- Neuronas de estimación de tiempo: son las neuronas que se encargan de estimar el tiempo que cada submovimiento le tomo al otro modelo realizar.
- Motoneuronas: son las neuronas que se encargan traducir la matriz de secuencia de coordenadas esféricas en las rotaciones sobre el modelo utilizando las librerías proporcionadas por el motor físico Havok.

6.4.1. Neuronas de estimación de movimiento

Como primer punto, como puede verse en la sección 6.2, es necesario poder realizar una conversión de coordenadas cartesianas (.x, y, z), a coordenadas esféricas (P, φ, θ), para esto existe principalmente las siguientes ecuaciones matemáticas:

Figura 59. **Coordenadas esfericas y cartesianas**

$$r = \sqrt{x^2 + y^2 + z^2} \qquad \theta = \begin{cases} \arctan\left(\frac{\sqrt{x^2+y^2}}{z}\right) & z > 0 \\ \frac{\pi}{2} & z = 0 \\ \pi + \arctan\left(\frac{\sqrt{x^2+y^2}}{z}\right) & z < 0 \end{cases} \qquad \varphi = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \frac{\pi}{2}\mathrm{sgn}(y) & x = 0 \\ \pi + \arctan\left(\frac{y}{x}\right) & x < 0 \end{cases}$$

Fuente: http://es.wikipedia.org/wiki/Coordenadas_esf%C3%A9ricas. Consulta: 15 de diciembre de 2011.

Con las ecuaciones anteriores es posible obtener a partir de un punto x, y, z en el plano cartesiano, la misma posición en dicho plano pero utilizando un radio, y los angulos φ y θ.

Ya que el objetivo es que la red neuronal genere la misma secuencia de movimientos con coordenadas esféricas, primero hay que definir cómo funcionará cada neurona.

Para las neuronas de estimación de movimiento, lo recomendable es que cada vértice sea procesado por una neurona, así para un antebrazo se terndría una neurona, para un muslo otra neurona, para la cadera otra neurona, y así sucesivamente, y de manera más general, que cada neurona procese la secuencia de movimiento de cada vértice, poniendo a cada neurona en la necesidad de tomar de cada matriz resultante el conjunto de coordenadas x, y, z, para dicho vértice, representando esta secuencia la suma ponderada de las entradas.

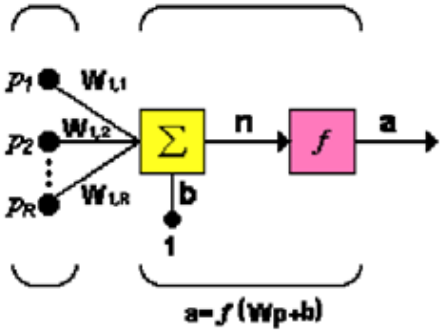
De tal forma que se tendría cada neurona con el siguiente diseño:

Figura 60. **Diseño de neurona**

P1: el punto x,y,z para el vértice $V_{k,}$ para la matriz resultante 1, en el frame 0

P2: el punto x,y,z para el vértice $V_{k,}$ para la matriz resultante 2, en el frame j

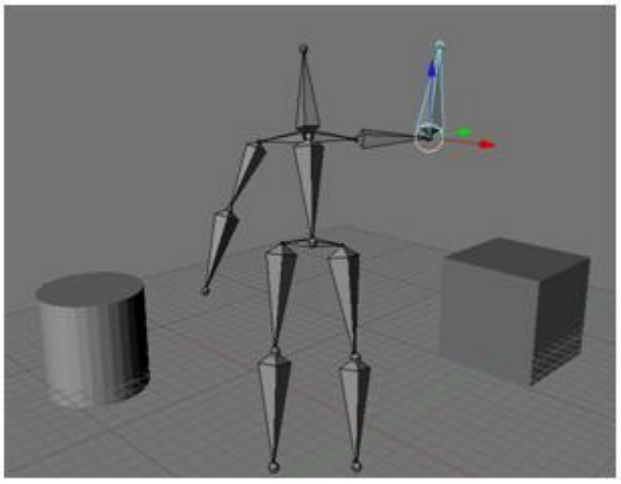
PR: el punto x,y,z para el vértice $V_{k,}$ para la matriz resultante R, en el frame j+n



Fuente: http://sav.us.es. Consulta: 15 de diciembre de 2011.

Al observar un hueso dentro de un esqueleto es posible reconocer que la longitud de este hueso permanece constante, así que de las tres coordenadas esféricas r, φ, θ, la que permanece constante es r, siempre para cada hueso, por lo que esta longitud no es de importancia al momento de rotar algún hueso dentro de algún esqueleto.

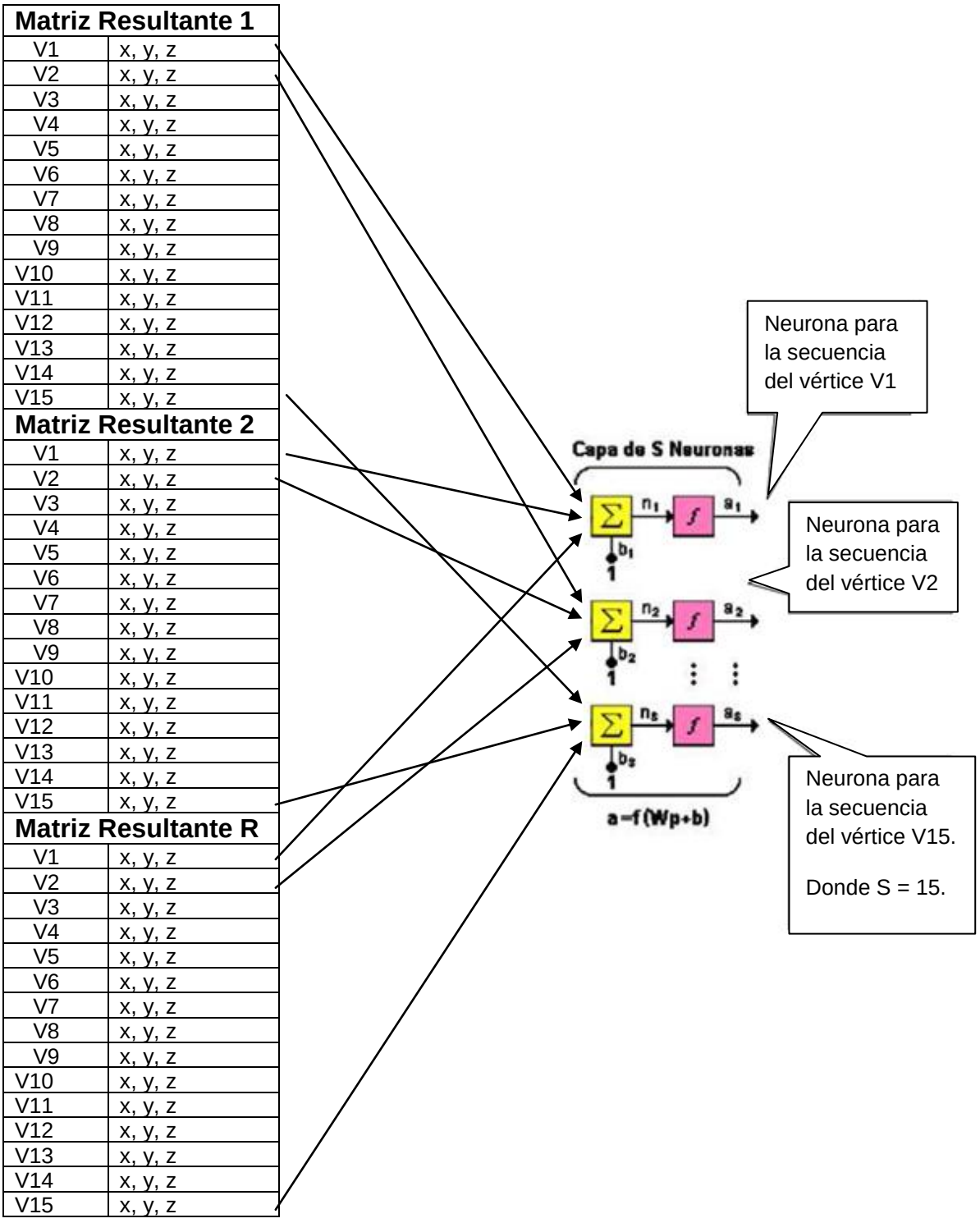
Figura 61. **Esqueleto en HAVOK**



Fuente: elaboración propia.

Por consiguiente el diseño de la red neuronal implicaría que para la capa de estimación de movimiento, el número de entradas de cada neurona sería igual al número de matrices resultantes de la secuencia de movimiento a imitar, es decir que se tendría una neurona que procese la secuencia de movimiento por cada vértice, esto se puede ver en la siguiente imagen.

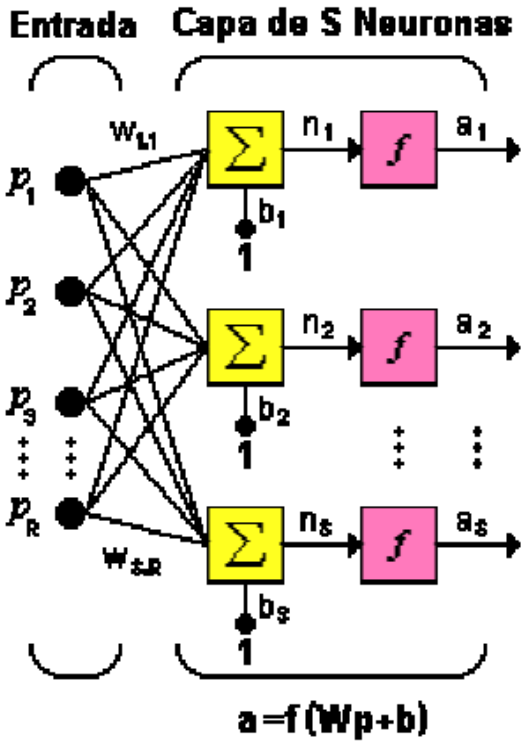
Figura 62. **Secuencia de movimiento para cada vértice**



Fuente: elaboración propia.

De la imagen anterior se obtiene entonces el siguiente diseño:

Figura 63. **Diseño de perceptron**



Fuente: elaboración propia.

Donde:

- Pi: entrada a la neurona
- Wi,j: peso sináptico para la conexión entre la neurona i, y la neurona j de la capa anterior
- b: es una ganancia o pérdida aplicada a la sumatoria ponderada, este valores común en redes neuronales tipo perceptrón para obtener un aprendizaje más acelerado, su valor inicial es aleatorio igual que los pesos sinapticos.

- n: es la suma ponderada total
- a: es la salida de la neurona que está en función de n

Y donde:

- R: Cantidad de matrices resultantes.
- S: Cantidad de vértices del modelo a imitar

Y la salida de la neurona quedaría de la siguiente manera:

$$a = F(w*(x,y,z) + b) = W*Esféricas(x,y,z) + b$$

6.4.2. Neuronas de estimación de tiempo

Estas neuronas como tales, son las encargadas de estimar el tiempo necesario para cada sub movimiento que se debe utilizar, al igual que lo hizo el modelo a imitar, es decir por ejemplo, muy diferente sería correr que caminar, pues caminar, puede ser interpretado como caminar lento, así las neuronas de estimación de tiempo, conforme vaya cambiando sus pesos sinápticos, el movimiento se irá acercando en cuestión de tiempo a lo que hizo el otro modelo.

Ya que uno de los aspectos del modelo perceptrón multicapa es que a partir de ciertas entradas, se espera que las salidas se vayan acercando a un valor conocido.

Para que el aprendizaje sea supervisado, las entradas a las neuronas de estimación de tiempo no pueden ser en si, el tiempo necesario para ejecutar la acción, sino un estimado de este, tal como ocurre en la naturaleza, por ejemplo se puede definir la siguiente clasificación:

Movimiento de tiempo corto: menos de un segundo, es decir que para una frecuencia de 60 frames por segundo, el movimiento tomo menos de sesenta frames en completarse.

Movimiento de tiempo medio: más de un segundo y menos de tres segundos, es decir que para una frecuencia de 60 frames por segundo, el movimiento tomo mas sesenta frames y menos de ciento ochenta frames en completarse.

Movimiento de tiempo largo: más de tres segundos, es decir que para una frecuencia de 60 frames por segundo, el movimiento tomo más de ciento ochenta frames en completarse.

De lo anterior se obtiene que para cada matriz resultante y para cada vértice, se le asigna un valor numérico estimado de tiempo, en base a los criterios antes descritos, por ejemplo, tiempo corto sería 30 frames, tiempo medio sería 120, tiempo estimado sería 240.

Tabla XXVI. **Matrices resultantes**

Matriz Resultante		
Vértice	Coordenadas	Tiempo Estimado
V1	x, y, z	Corto
V2	x, y, z	Medio
V3	x, y, z	Largo
V4	x, y, z	Medio
V5	x, y, z	Largo
V6	x, y, z	Medio
V7	x, y, z	Corto
V8	x, y, z	Largo
V9	x, y, z	Medio
V10	x, y, z	Largo
V11	x, y, z	Corto
V12	x, y, z	Largo
V13	x, y, z	Medio
V14	x, y, z	Corto
V15	x, y, z	Largo

Matriz Resultante		
Vértice	Coordenadas	Tiempo Estimado
V1	x, y, z	30
V2	x, y, z	120
V3	x, y, z	240
V4	x, y, z	120
V5	x, y, z	240
V6	x, y, z	120
V7	x, y, z	30
V8	x, y, z	240
V9	x, y, z	120
V10	x, y, z	240
V11	x, y, z	30
V12	x, y, z	240
V13	x, y, z	120
V14	x, y, z	30
V15	x, y, z	240

Fuente: elaboración propia.

Y por consiguiente la salida de la neurona para estas matrices sería la siguiente:

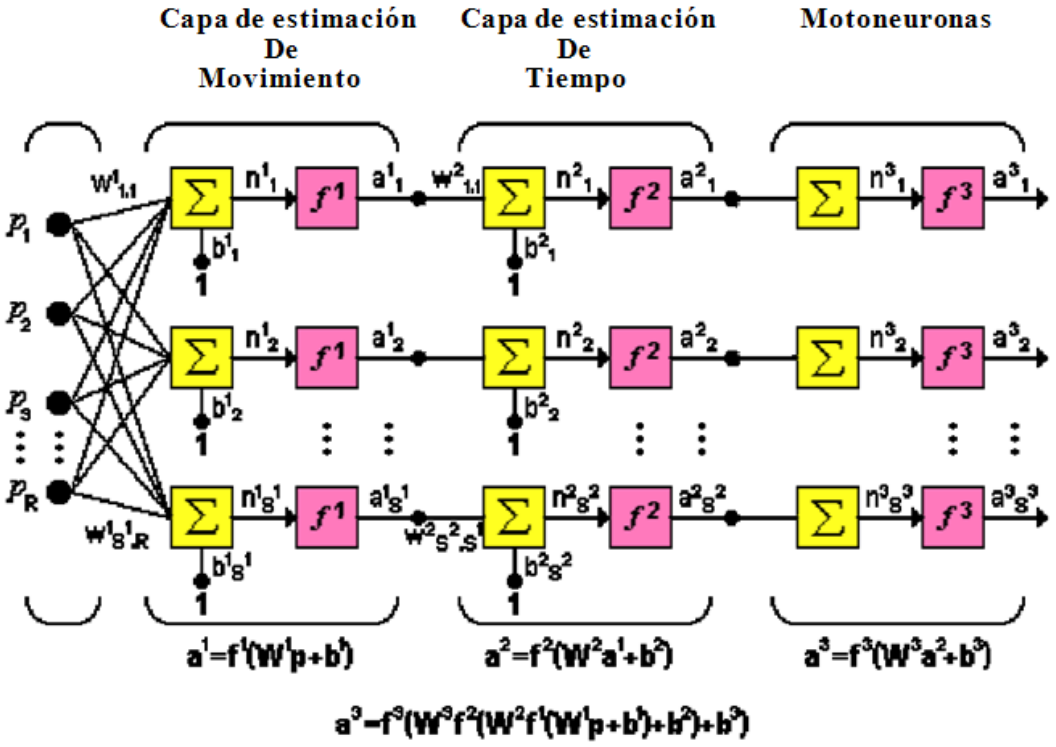
$$a = F(WTi + b) = WTi + b$$

Y conforme W va cambiando, el tiempo del movimiento se va acercando al tiempo necesario para poder imitar aceptablemente al otro modelo.

6.4.3. Diseño de la red

Del anterior análisis se puede obtener el siguiente diseño de la red neuronal:

Figura 64. **Diseño de red**



Fuente: elaboración propia.

De la imagen anterior puede verse la presencia de motoneuronas, estas neuronas son descritas más adelante.

Donde el proceso de aprendizaje para el movimiento queda definido de la siguiente manera:

- Si $\Phi_{esperado} = \Phi_i$ y $\theta_{esperado} = \theta_i$; y (Φ_a != Φ_i ó θ_a != θ_i);
  entonces $W_{nuevo} = W_{anterior} + (\Phi_i - \Phi_a + \theta_i - \theta_a)$
  y $b_{nuevo} = b_{anterior} + (\Phi_i - \Phi_a + \theta_i - \theta_a)$

- Si $\Phi_{esperado} = \Phi_i$ y $\theta_{esperado} = \theta_i$; y ($\Phi_i = \Phi_a$ y $\theta_i = \theta_a$);
  entonces $W_{nuevo} = W_{anterior}$
  y $b_{nuevo} = b_{anterior}$

- Donde a = F(w*(x,y,z) + b) = W*Esféricas(x,y,z) + b

Y donde el proceso de aprendizaje para el tiempo queda definido de la siguiente manera:

Si $T_{esperado} = T_i$; y (T_a != T_i);
entonces $W_{nuevo} = W_{anterior} + (T_{esperado} - T_a)$
y $b_{nuevo} = b_{anterior} + (T_{esperado} - T_a)$

- Si $T_{esperado} = T_i$; y ($T_a = T_i$);
  entonces $W_{nuevo} = W_{anterior}$
  y $b_{nuevo} = b_{anterior}$

Al momento de recalcular los pesos, este proceso de aprendizaje debe ser evaluado por cada punto resultante dentro de la secuencia que cada neurona procesa, lo cual acelera fuertemente el aprendizaje de la red neuronal.

A continuación se presenta la definición de clases para la implementación de la red neuronal.

6.5. Definición de clases

A continuación se presenta la descripcion de las clases involucradas, asi como sus metodos, atributos principales y relevantes que son necesarios para el desarrollo de la red neuronal propia de este trabajo de investigación.

6.5.1. Clase neurona

Con esta clase, ya es posible crear neuronas para estimar movimiento y tiempo, tanto para la clase NeuronaTiempo y NeuronaMovimiento, basta con que hereden de Neurona y en ellas sea redefinida la función a = F(wp+b).

Figura 65. **Diseño de clase neurona**

Neurona
Attributes
- id : int - Salida : salida - SalidaDeseada : salida - Entradas : Vector - NeuronasCapaSiguiente : Vector - idVertice : int
Operations
+ ActualizarSalidaDeseada(SalidaDeseada : void) : Void + GetSalida( ) : salida + SetEntrada(Entrada : entrada, idNeuronaCapaAnterior : int) : void + a(entrada : Secuencia) : void

Fuente: elaboración propia.

Dentro de los métodos y atributos más importantes a implementar están:

Tabla XXVII. **Descripción de atributos clase neurona**

Atributo	Descripción
Id	Identificador de la neurona.
Salida	Valor de salida en la neurona que se actualiza cada vez que se manda a llamar a su función a = F(wp+b)
SalidaDeseada	Es el valor que se espera tenga como salida después de evaluar su función a = F(Wp+b)

Fuente: elaboración propia.

Tabla XXVIII. **Otros atributos de clase neurona**

Entradas	Vector de entradas, este vector forma la secuencia del vértice que tiene asignada la neurona.
NeuronasCapaSiguiente	Contiene las neuronas a las cuales se va a pasar el valor de salida.
idVertice	Identificador del vértice del cual la neurona tiene que procesar su secuencia.

Fuente: elaboración propia.

Tabla XXIX. **Métodos de clase neurona**

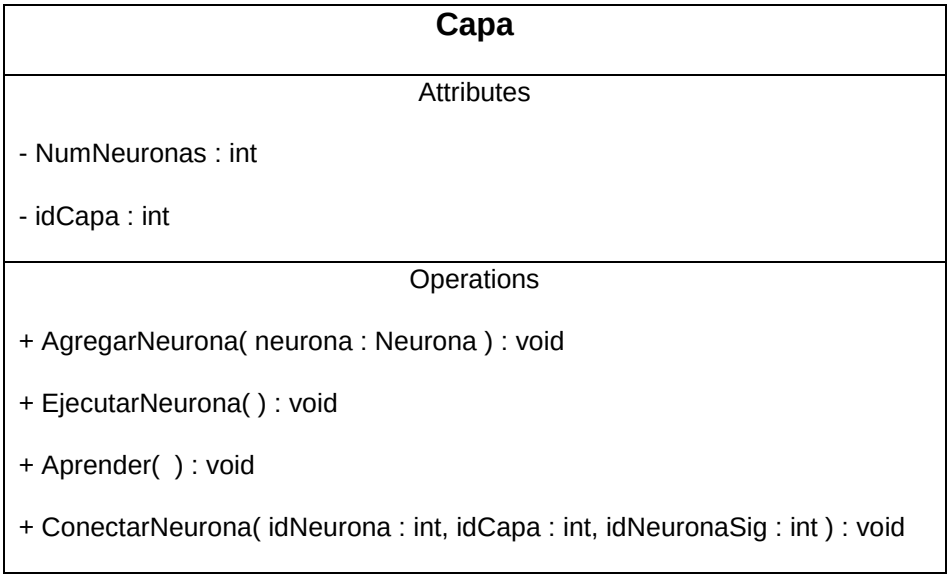
Método	Descripción
ActualizarSalidaDeseada(SalidaDeseada:void) : void	Actualiza las salidas deseadas de la neurona,
getSalida(): Salida	Valor de salida en la neurona que se actualiza cada vez que se manda a llamar a su función a = F(wp+b)
SetEntrada(Entrada:entrada, idNeuronaCapaAnterior : int) : void	Inicializa las entradas de la neurona a partir de la capa especificada
a(entrada : Secuencia) : void	Función de generación de salidas a partir de la secuencia especificada como parámetro.

Fuente: elaboración propia.

6.5.2. Clase capa

Descripción: La clase capa agrupa las neuronas de un solo tipo y su función principal es organizarlas y asociarlas con neuronas de otra capa, permitiento la correcta y ordenada comunicación entre ellas.

Figura 66. **Diseño de clase capa**



Fuente: elaboración propia.

Dentro de los métodos y atributos más importantes a implementar están:

Tabla XXX. **Atributos clase capa**

Atributo	Descripción
NumNeuronas	Cantidad de neuronas de la capa.
idCapa	Identificador de la capa.

Fuente: elaboración propia.

Tabla XXXI. **Métodos clase capa**

Método	Descripción
AgregarNeurona(neurona : Neurona) : void	Agrega una nueva neurona a la capa,
EjecutarNeuronas( ) : void	Valor de salida en la neurona que se actualiza cada vez que se manda a llamar a a su función a

Fuente: elaboración propia.

6.5.3. Clase red

Descripción: La clase red agrupa las capas de neuronas, y su función principal es organizarlas y asociarlas entre si, instanciando así la plataforma para el procesamiento de la información.

Figura 67. **Diseño de clase red**

Red
Attributes
- idRed : int
- Capas : vector
- entradas : Secuencia
- Salida : secuenciaEsferica
Operations
+ getCapaEntrada( ) : capa
+ getCapaSalida( ) : capa
+ getCapaInterna( ) : capa
+ ConectarCapa(idOrigen : int, idDestino : int) : void
+ CrearCapas( ) : void
+ SetEntradas(secuencia : secuencia) : void
+ getSalida( ) : SecuenciaEsferica
+ Aprender( ) : void

Fuente: elaboración propia.

Dentro de los métodos y atributos más importantes a implementar están:

Tabla XXXII. **Atributos de clase red**

Atributo	Descripción
idRed	Identificador de la red
Capas	Vector de capas de la red
Entradas	Secuencia de movimiento
Salida	Secuencia esférica de movimiento

Fuente: elaboración propia.

Tabla XXXIII. **Métodos de clase red**

Método	Descripción
getCapaEntrada() : capa	Devuelve la capa de entrada que es también la capa de estimación de movimiento
getCapaSalida(): capa	Devuelve la capa de salida que es la capa de motoneuronas
getCapaInterna(): capa	Devuelve la capa de interna que es la capa de estimación de tiempo.
ConectarCapa(idOrigen : int, idDestino : int) : void	Conecta una capa con otra capa de la red especificada.
CrearCapas() : void	Crea las tres capas que constituyen la red.
SetEntradas(secuencia: Secuencia): void	Inicializa las entradas en la primera capa de la red
getSalida(): SecuenciaEsferica	Devuelve la secuencia con coordenadas esféricas para imitar al modelo observado.
Aprender(): void	Ejecuta los algoritmos de aprendizaje de las capas de la red en el orden correspondiente.

Fuente: elaboración propia.

6.6. El robot discípulo y los objetos del entorno

La importancia de la interacción con objetos inanimados dentro de un entorno virtual es fuertemente significativa al momento de implemenentar el aprendizaje por imitación, por esta razón, esta es una parte donde la SDK de Havok, ahorra gran parte del trabajo.

La SDK de Havok, una tecnología middleware para implementar simulaciones rápidas y en tiempo real de cuerpos rígidos con todas las restricciones y consecuencias implicadas en el mundo real. Havok se puede utilizar en aplicaciones en las que los objetos deben interactuar dentro de un espacio 3D de forma realista.

Un motor fisico tiene tres tareas básicas a realizar en su bucle principal de simulación:

A. Determinar qué objetos se superponen o se solapan unos a (por ejemplo, una nave espacial ha chocado con un hangar de la pared, o con otra nave espacial).

B. Resolver todas las fuerzas que actúan sobre los objetos. Algunas fuerzas se producen como consecuencia de la detección de colisiones (por ejemplo, dosnaves espaciales chocando entre sí), otros debido a la entrada desde fuera de la simulación (por ejemplo, la gravedad del medio ambiente o la fuerza impulsora de los motores de un barco).

C. Después de haber reunido todas las fuerzas, la simulación avanza por el paso de tiempo de simulación y el nuevo estado de los objetos se calcula(posición, orientación, velocidad, aceleración, etc) . Esta

información se utiliza para actualizar las visualizaciones correspondientes de los objetos, mantener toda la lógica del juego.

Es decir que el motor físico de Havok se encargara de re calcular todas las consecuencias físicas de la interacción dentro del entorno virtual, además de que se proporcionan las herramientas para conocer la información de dichas consecuencias.

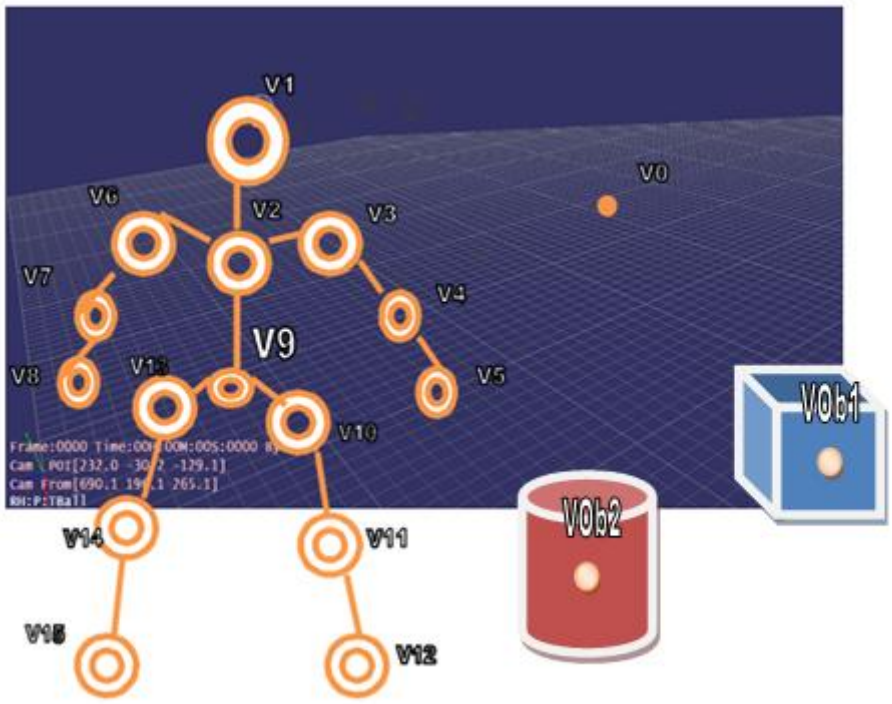
Por ejemplo para la detección de colisiones entre objetos, es Havok quien se encarga de ejecutar las consecuencias físicas de dichas colisiones dentro de la simulación. La detección de colisiones es uno de los núcleos de cualquier motor físico. Es notable que esta gestión consume una gran parte del tiempo de CPU requerido para la simulación de juego.

Además de todo esto, Havok posee mecanismos para mejorar la eficiencia de todos estos procesos de simulación física, que optimizan el desempeño de la aplicación.

6.6.1. La red neuronal y la interacción con los objetos del entorno

Para poder implementar el aprendizaje por imitación interactuando con otros objetos, será necesario incluir esto dentro de la red neuronal del robot discípulo. Al igual que como se vio en la sección 5.2, es necesario ver al objeto inanimado que esta agregado o presente dentro de la simulación como otro vértice mas, respecto del vértice 0.

Figura 68. **Entorno**



Fuente: elaboración propia.

Y de igual forma, se agregan estos vértices a la matriz de posición.

Tabla XXXIV. **Matriz de posicion**

Vértice	Vértice de Referencia	X	Y	Z
V1	V2	54.35826279	95.236054	53.3851887
V2	V9	22.85689793	30.8521964	90.8232256
V3	V2	8.335508006	90.1606397	14.7333883
V4	V3	8.365976397	11.5340253	15.7803157
V5	V4	82.03450214	64.8241122	64.2154763
V6	V2	37.30181535	50.6462965	50.0843834
V7	V6	62.76749901	25.9664013	81.4933261
V8	V7	9.712110769	61.6875929	62.617009
V9	V0	29.25703303	14.9611387	12.6512228
V10	V9	31.46522487	38.7310019	14.5983226
V11	V10	50.00706045	14.1918603	88.5106598
V12	V12	12.90813689	38.7213429	98.7667775
V13	V9	0.941653737	62.5543672	47.9014826
V14	V13	9.009364846	56.8306951	47.5874306
V15	V14	34.56158605	38.7185153	24.9096831
VOb1	V0	31.46522487	38.7310019	14.5983226
VOb2	V0	22.85689793	30.8521964	90.8232256

Fuente: elaboración propia.

Asimismo es necesario agregar una neurona por cada objeto que esté presente en la simulación, el diagrama de la red neuronal permanece únicamente con las siguientes variaciones:

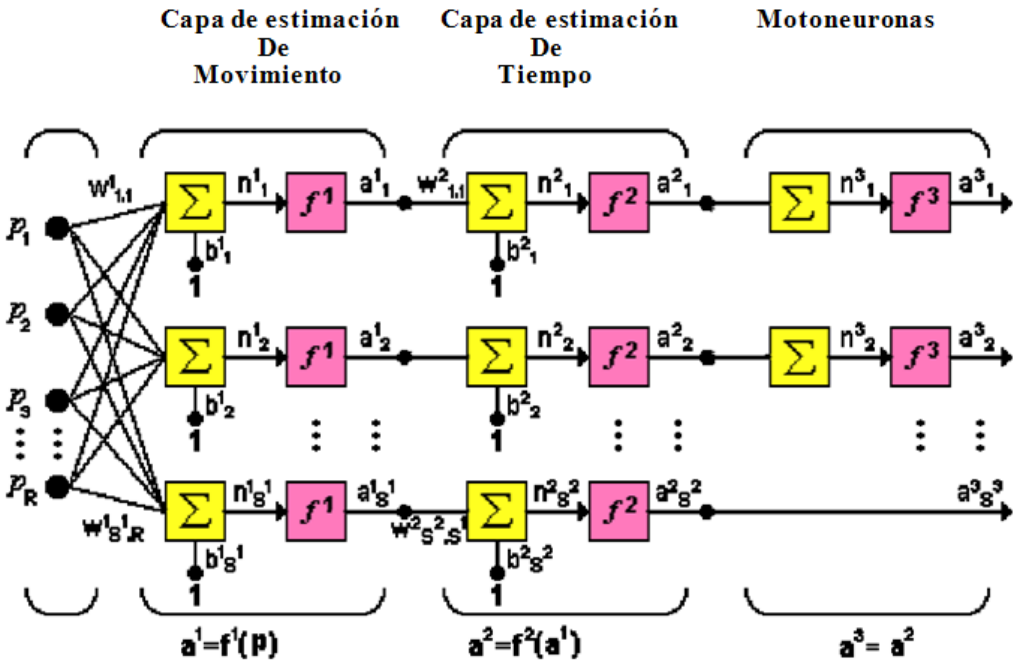
- Pr es la entrada de un vértice que representa un objeto dentro del entorno virtual

- Se asigna una neurona para almacenar la información de la secuencia del movimiento del objeto, asi como otra neurona para almacenar el

tiempo que le tomo al objeto cambiar de posiciones en su secuencia, si es que hubieron cambios durante el movimiento.

- No se asigna motoneurona para el objeto porque es ajeno al robot discípulo, por lo que la salida de esta ultima capa es igual a la capa de estimación de tiempo.

Figura 69. **Diseño de red neuronal**



Fuente: elaboración propia.

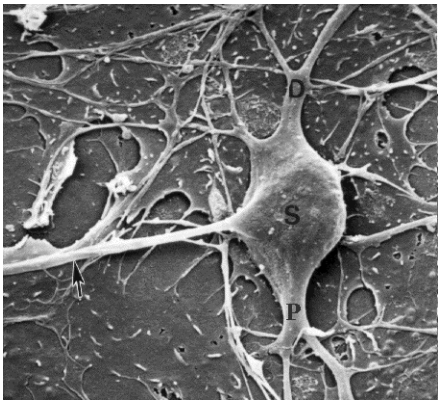
En otras palabras, la red neuronal solo se encargara de comparar finalmente la posición final resultante de los objetos del entorno respecto de lo esperado a partir de los movimientos del robot maestro, y así indicar si la interacción y/o aprendizaje han sido correctos respecto del resto del entorno.

7. LAS MOTONEURONAS

7.1. ¿Qué es una motoneurona?

Las motoneuronas o neuronas motoras, se encuentran entre las más grandes de las células nerviosas. Comienzan desde el cerebro hasta la médula espinal y desde la médula van hacia los músculos. Su función es transmitir señales y órdenes del cerebro a los músculos. Las órdenes se transmiten a los músculos siguiendo dos clasificaciones de motoneuronas: las motoneuronas centrales que van del cerebro a la médula espinal y las motoneuronas periféricas que van de la médula espinal a los músculos, biológicamente estas neuronas proyectan las salidas de sus axones fuera del sistema nervioso central para poder estimular los músculos del cuerpo, a continuación se muestra una imagen real de una motoneurona obtenida por medio de micrografía electrónica.

Figura 70. **Motoneurona biológica**



Fuente: http://reocities.com/FashionAvenue/agency/1429/biologia.htm. Consulta: 15 de diciembre de 2011.

Las neuronas motoras que formen parte de la red neuronal especular artificial, para el diseño que se describe en este documento, caen dentro de las dos clasificaciones de neuronas motoras, es decir que son tanto motoneuronas centrales como motoneuronas periféricas, ya que estas neuronas controlarán directamente los movimientos del modelo humanoide.

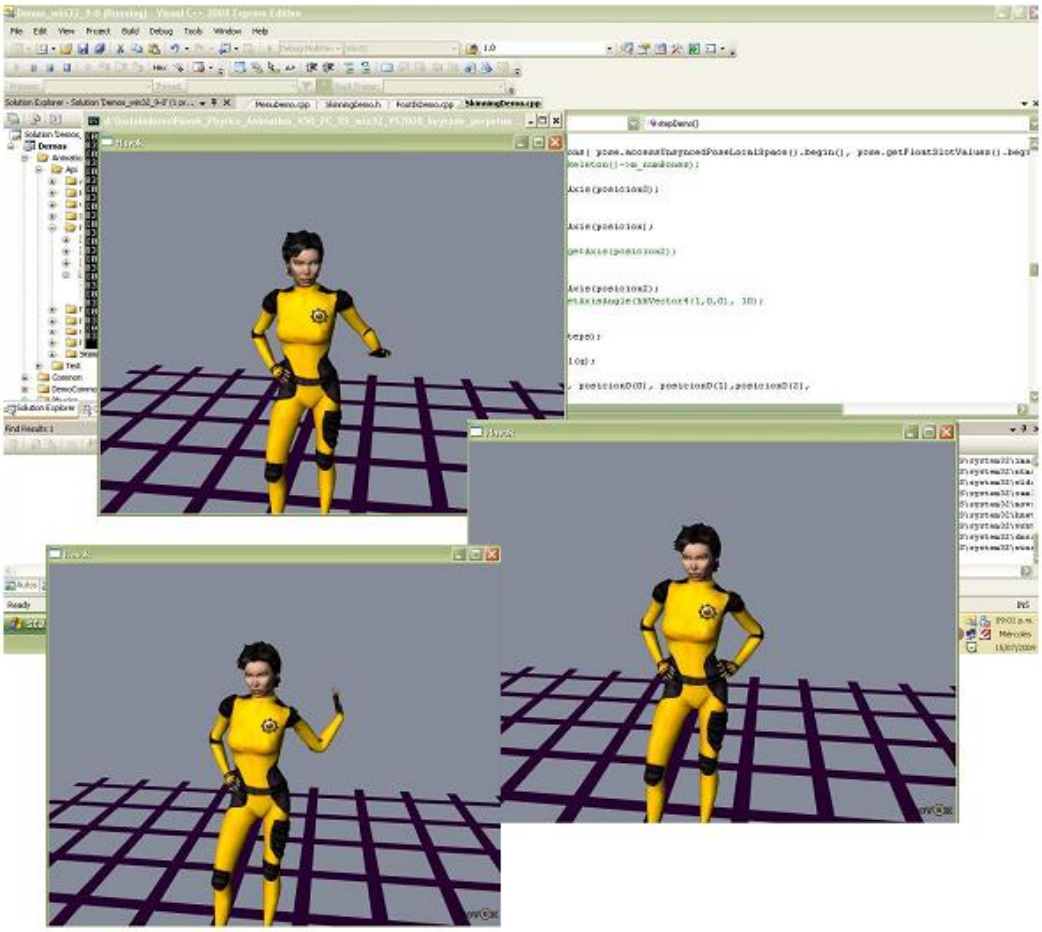
Estas motoneuronas se encargarán de procesar una secuencia de movimientos en ángulos sobre los vértices del modelo, siguiendo estrictamente el tiempo estimado de la red neuronal, y en cada frame irá modificando la posición de cada hueso, este cambio de posición estará definido por una variable encargada de controlar el impulso nervioso sobre las extremidades del modelo. A continuación se describe el diseño de las motoneuronas de la red neuronal especular.

7.2. Las motoneuronas con base en HAVOK

Para poder definir estrictamente el diseño de las neuronas, primero es necesario entender que clases provistas por el motor Havok, son las más adecuadas para formar parte de la motoneurona.

Después de investigar y de realizar varias pruebas sobre los ejemplos provistos por el SDK Havok, se logró la rotación de los huesos del brazo de una animación sobre el esqueleto y modelo de una mujer, como se ve en la siguiente imagen.

Figura 71. **Animación de mujer en HAVOK**



Fuente: elaboración propia.

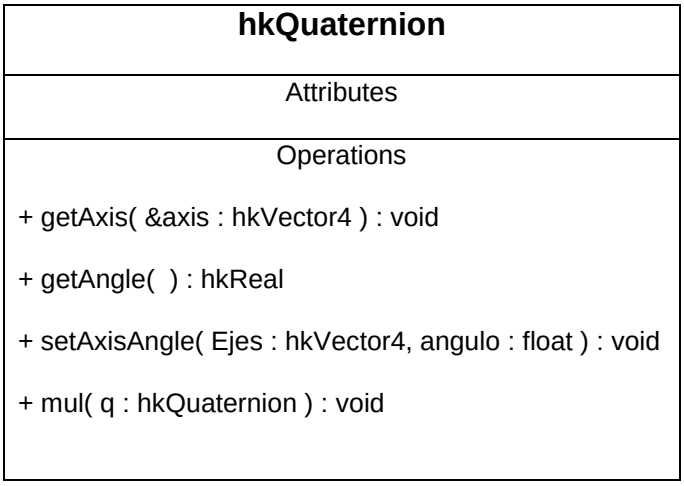
7.2.1. Clases y métodos a utilizar

Nuevamente son las clases hkVector4, hkaPose y hkQuaternion las necesarias para realizar la rotación de huesos sobre un esqueleto, los métodos necesarios y su utilización para este propósito se describen a continuación.

Para poder aplicar rotación a un hueso dentro de un esqueleto en el entorno Havok, puede verse el siguiente código:

```
hkQuaternion q;
q.setAxisAngle( hkVector4(1,0,0), 10);
pose.accessBoneLocalSpace(id_Hueso).m_rotation.mul(q);
```

Figura 72. **Diseño de clase hkQuaternion**



Fuente: elaboración propia.

- Primero se define el objeto q de tipo hkQuaternion

- Se define la rotación por medio del método setAxisAngle enviándole como parámetro un vector que debe tener un valor 1 en el eje sobre el cual se quiere aplicar la rotación, y se envía también como parámetro el ángulo sobre el cual se quiere posicionar el hueso en los ejes especificados por el vector que se envía como parámetro.

- La variable pose que es de tipo hkaPose y que hace referencia al esqueleto sobreel cual se quiere aplicar la rotación, provee mediante el método accessBoneLocalSpace acceso al objeto m_rotation de tipo hkQsTransform, asociado al hueso especificado como parámetro.

- m_rotation provee el método mul que aplica la rotación espeficiada por el objeto q.

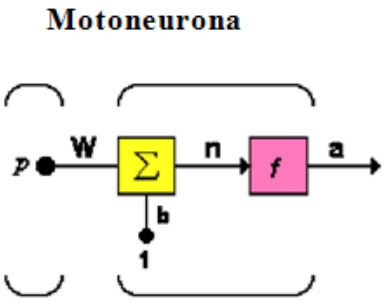
Sin embargo la neurona en cada frame debe modificar el valor del ángulo resultante con el código q.setAxisAngle(hkVector4(1,0,0), angulo); a manera que se imite el proceso mental de flujos de pulsos eléctricos sobre los músculos, y al mismo tiempo se mide el tiempo invertido, y la animación es continua al que la ve, para que no vea solo saltos bruscos de movimiento, esto se describe a continuación.

7.2.2. El diseño de la motoneurona

Cada motoneurona procesa toda una secuencia de movimientos para un vértice especifico, tomando en cuenta solo los ángulos φ, θ sin el radio, ya que este representa la longitud del hueso que permanecerá constante en todo tiempo.

Figura 73. **Diseño de motoneurona**

Vértice	Coordenadas	Tiempo Estimado en Frames
V1	r. φ, θ	53
V1	r. φ, θ	90
V1	r. φ, θ	14
V1	r. φ, θ	15
V1	r. φ, θ	64
V1	r. φ, θ	50
V1	r. φ, θ	81
V1	r. φ, θ	62
V1	r. φ, θ	12
V1	r. φ, θ	14
V1	r. φ, θ	88
V1	r. φ, θ	98
V1	r. φ, θ	47
V1	r. φ, θ	47
V1	r. φ, θ	24



Neurona

Attributes

- id : int
- Salida : salida
- SalidaDeseada : salida
- Entradas: Vector
- NeuronasCapaSiguiente : Vector
- IdVertice : int

Operations

+ ActualizarSalidaDeseada(SalidaDeseada : Void) : void
+ GetSalida( ) : salida
+ SetEntrada(Entrada : entrada, idNeuronaCapaAnterior : int) : void
+ a(entrada : Secuencia) : void

Fuente: elaboración propia.

De igual manera, la clase motoneurona hereda de la clase Neurona, pero lo que identifica a la neurona como motoneurona, es que su función a = F (Secuencia) se va ejecutando conforme se va avanzando en la animación, es

decir que para realizar la primer rotación de la imagen anterior se tendría que realizar el siguiente pseudocódigo.

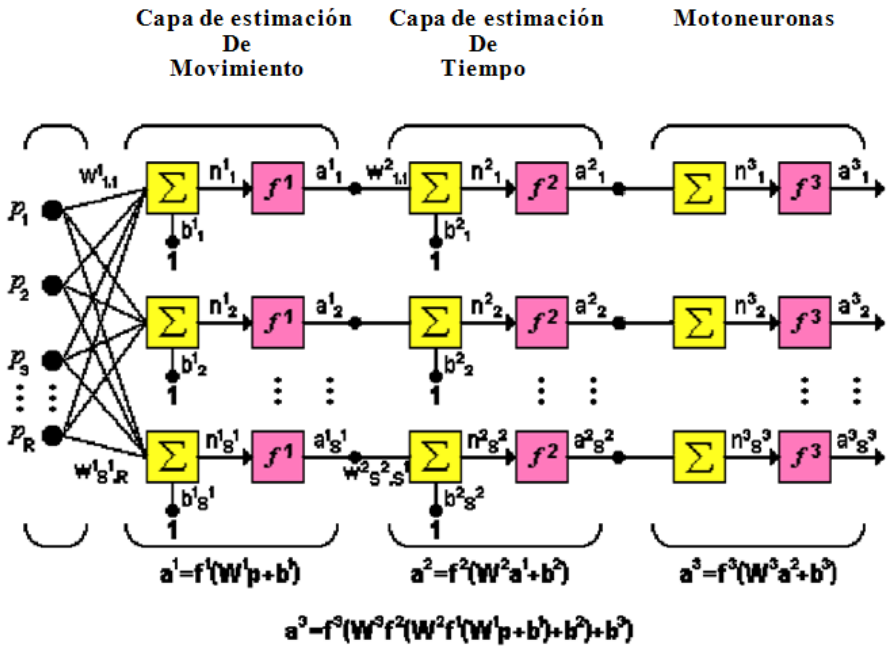
Se obtiene el tiempo estimado. Se sabe que tarda 53 frames en realizar la rotación (esto significa que realizara 53 submovimientos, uno en cada frame, para realizar toda la rotación.

A. Se obtiene la posición actual del hueso en términos de ángulo.
B. Se obtiene la diferencia de ángulos entre la posición actual y la posición buscada
C. Se divide la diferencia de ángulos en 53 partes (cada una de estas partes se llamará diferencial de ángulo).
D. Se suma o resta un diferencial de ángulo al ángulo actual para llegar a la posición buscada, dependiendo si la diferencia de ángulos es positiva o negativa.
E. Si existe otra posición que buscar se repite desde el paso A.

Por consiguiente en los algoritmos de aprendizaje es necesario especificar una bandera que se inicialice en verdadero cuando se acabe de generar la matriz de secuencia de coordenadas esféricas, para que las neuronas empiecen a enviar un flujo de movimiento a los huesos del modelo, consumiendo así toda la secuencia de posiciones, y esta bandera regresa a su valor original cuando se termine de traducir toda la matriz de secuencia en pulsos de movimiento.

Finalmente puede verse la siguiente imagen que represente entonces el diseño de la red especular.

Figura 74. **Diseño de red especular final**



Fuente: elaboración propia.

CONCLUSIONES

1. La capacidad de aprendizaje por imitación que un software puede lograr mediante la utilización de una red neuronal dentro de un entorno basado en animación 3D resulta ser bastante acercado al proceso natural del cerebro humano, la implementación del software descrito en este trabajo puede demostrar perfectamente que mediante inteligencia artificial, es posible que un modelo humanoide pueda imitar el comportamiento de otro de una forma bastante precisa. Sin embargo el cerebro humano realiza el aprendizaje tomando en cuenta todas las relaciones y asociaciones entre los objetos de su entorno, utilizando los conceptos que se han formado a lo largo de su vida.

2. Las neuronas asociadas al aprendizaje por imitación del cerebro, son llamadas neuronas especulares, estas presentan actividad cuando se está observando a otro ser similar a sí mismo, y también presentan actividad al momento de realizar movimientos similares, en otras palabras, estas neuronas tienen la capacidad de simular la realización de los movimientos vistos en otros seres similares, es por esto que el estudio de dichas neuronas es clave para el desarrollo de un software inteligente que aprenda basándose en la imitación.

3. La implementación del software inteligente descrito en este documento, reproduciría el experimento de Albert Bandura en un entorno basado en animación 3D, representando así un ejemplo de que la inteligencia artificial, asociada a la personalidad de los individuos, puede ser posible.

4. Un entorno tridimensional elimina la necesidad de poseer sensores inmensamente precisos para captar el mundo real, pues estos pueden ser modelados y simulados y reprogramados, lo que permite un avance más fluido en la investigación de inteligencia artificial.

5. El modelo perceptrón multicapa como modelo base para el diseño de la red neuronal especular artificial, resulta ser el más adecuado no solo por ser un modelo supervisado y por que inicialmente fue conceptualizado para explicar estos procesos inteligentes, sino porque su diseño se adapta perfectamente a las necesidades implicadas para el aprendizaje por imitación.

6. Las redes neuronales especulares en este documento tienen como principales características la de que la información que procesan debe ser una secuencia de movimientos en un formato que se obtuvo del entorno en el que se encuentra, y transformar en cada entrenamiento esta secuencia de entrada en otra secuencia de salida, pero en términos de movimientos del modelo, que sea similar a lo que el modelo a imitar realizó, y su capa de salida debe estar formada por motoneuronas, encargadas de aplicar movimiento a las extremidades del modelo sobre el cual se ha asociado dicha red neuronal.

7. Una investigación complementaria al presente trabajo sería agregar a este aprendizaje por imitación las relaciones y asociaciones que el modelo humanoide necesita para entender su entorno, y así proveer al modelo humanoide de conciencia y personalidad artificial.

8. El motor físico Havok proporciona gran variedad de herramientas, librerías y clases para la investigación de la inteligencia artificial, porque elimina la necesidad por parte del desarrollador e investigador, de implementar el código necesario para controlar todas las implicaciones físicas de un entorno virtual, permitiendo enfocarse únicamente en la lógica que se desea aplicar.

9. Actualmente las computadoras son muy rápidas para procesar información, pero realizan sus acciones sin entender realmente el significado de tales operaciones, lo que no les permite cambiar el flujo de su acciones, mientras que el cerebro humano cambia su conducta de acuerdo a sus necesidades con respecto al entorno en el que se encuentra, esto está estrechamente relacionado con la personalidad del individuo, lograr que el software pueda entender lo que hace y si existe la necesidad de un cambio pueda cambiar su conducta, puede ser investigado entendiendo primero las teorías psicológicas existentes.

10. Conocer los procesos inteligentes del cerebro humano son clave para poder crear nuevo software que logre procesar información de manera más inteligente y más versátil, por lo que todas las investigaciones psicológicas son una fuente inmensa de información que puede inspirar nuevas ideas para la construcción e investigación de software cada vez más inteligente que responda a diferentes situaciones.

11. Muchas invenciones del ser humano se han inspirado en creaciones de la naturaleza, entender la versatilidad del cerebro humano es clave para unir dicha capacidad a la rapidez de las computadoras.

RECOMENDACIONES

1. Para la realización de una investigación más profunda, es recomendable buscar una mayor comunicación con los campos de investigación de la psicología, para la búsqueda de la implementación de nuevas aplicaciones inteligentes.

2. Muchas invenciones del ser humano han sido inspiradas en la naturaleza, lo que implica que la observación de procesos inteligentes tanto en humanos como animales puede inspirar nuevas ideas para la implementación de inteligencia artificial más versátil y dinámica.

3. De la manera como Albert Bandura investigó los procesos de aprendizaje por imitación por medio de experimentos nuevos, para la implementación de nueva inteligencia artificial se recomienda la experimentación dentro de un entorno virtual.

4. Se recomienda la investigación transmitir la conciencia de la necesidad de entender los procesos inteligentes versátiles que se tiene actualmente, que son los procesos del cerebro humano, para así poder implementar nueva inteligencia artificial.

5. Se recomienda leer detenidamente la documentación de Havok para poder implementar e investigar nuevas tecnologías de inteligencia artificial dentro de dicho entorno.

6. Sería de mucho beneficio para la docencia de la Universidad San Carlos de Guatemala la implementación del software descrito en este documento para la enseñanza de la inteligencia artificial y la animación 3D.

BIBLIOGRAFÍA

1. ARRECIS, Giovanni. *Blender, programación, tutoriale.* [en línea]. Texto ed. 1.0. [Guatemala]: 2010. [ref. 10 de noviembre de 2010]. Disponible en Web: <http://arrecis.wordpress.com/about/.>

2. BOEREE, Jorge. *Teorías de la personalidad*. [en línea]. Texto ed. 1.0. [Estados Unidos de América]: 2002. [ref. 3 de enero de 2010]. Disponible en Web: <http://www.psicologia-online.com/ebooks/personalidad/bandura.htm.>

3. FLORES, Jesús. *Las bases del aprendizaje*. [en línea]. Texto ed. 1.0. [España]: 2005. [ref. 12 de enero de 2009]. Disponible en Web: <http://www.down21.org/salud/neurobiologia/bases_aprend.htm.>

4. LUZARDO, Gonzalo. *Inteligencia artificial en ambientes virtuales*. [en línea]. Texto ed. 1.0. [España]: 2009. [ref. 15 de junio de 2009]. Disponible en Web: <http://blog.espol.edu.ec/gluzardo/files/2009/03/trabajo-final.pdf.>

5. GUIN, Luis. *Neuronas*. [en línea]. Texto ed. 1.0. [Estados Unidos de América]: 2001. [ref. 06 de junio de 2009]. No disponible en Web: <www.geocities.com/luiguin_web/neuronas.htm.>

6. HAVOK. *Havok_Physics_Animation_650_Pc_Xs_User_Guide*. [en línea]. Texto ed. 1.0. [Estados Unidos de América]: 2009. [ref. 06 de junio de 2009]. Disponible en Web: <http://software.intel.com/sites/havok/.>

7. _____. Havok_Physics_Animation_650_Pc_Xs_Quickstart_Guide. [en línea]. Texto ed. 1.0. [Estados Unidos de América]: 2009. [ref. 06 de junio de 2009]. Disponible en Web: <http://software.intel.com/sites/havok/.>

8. MATUTE, Helena. *Del aprendizaje natural al aprendizaje artificial*. [en línea]. Texto ed. 1.0. [España]: 2004. [ref. 10 de junio de 2009]. Disponible en Web: <http://paginaspersonales.deusto.es/matute/psicoteca/articulos/Matute04.htm.>

9. Universidad de la Rioja. *Havok*. [en línea] texto ed. 1.0. [España]. 2009. [ref. 3 de Mayo de 2011]. Disponible en Web: <https://belenus.unirioja.es/~igmiro/web/havok.html.>

10. Universidad Politécnica de Madrid. *Artificial neural networks*. [en línea] texto ed. 1.0. [Madrid, España]. Página oficial, 2008. [ref. 5 de junio de 2010]. Disponible en Web: <http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm.>

11. Wikipedia. *Perceptrón multicapa*. [en línea] texto ed. 1.0. [Estados Unidos de América]. 2005. [ref. 10 de mayo de 2010]. Disponible en Web: <http://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa.>

ANEXO: EL ENTORNO HAVOK

Dentro de las herramientas analizadas para la implementación del software del que se hace énfasis en este trabajo de investigación, se ha seleccionado el motor físico Havok.

Havok (Havok Physics) es un motor físico middleware desarrollado por una compañía irlandesa con el mismo nombre, que por medio de simulación dinámica permite recrear interacciones de carácter físico en videojuegos y otras aplicaciones multimedia. Recientemente este motor físico fue adquirido por Intel y liberado bajo una licencia que permite su libre uso para fines no comerciales o educativos.

La aparición de Havok fue en la Gamers Developer Conference en el 2000, en la versión 1.0, que para hoy su versión más actual es la 2011.2.0 lanzada para Visual Studio 2010.

Su código fuente está basado en C/C++ por lo posee compatibilidad con cualquier sistema con un compilador C/C++ válido. Este motor físico sido trasladado a Windows, Unix, MacOs X, así como a plataformas de videojuegos por consolas Xbox, Xbox360, Nintendo GameCube, Playstation2, Playstation3, PlaystationPortable (PSP), y Nintendo Wii.

La facilidad de utilización y versatilidad de sus fuentes, así como sus posibilidades le permitieron a este motor físico ser aceptado rápidamente en la industria del videojuego y licenciado para gran mayoría de videojuegos

que requerían un motor gráfico, consiguiendo ser percibido prácticamente en la única alternativa, antes de la aparición de su competidor más directo, Ageia PhysX.

La aceptación de Havok es tal que su implementación en construcción de videojuegos a nivel comercial forma parte de muchos títulos del genero conocidos hoy en día, entre los cuales figuran: Age of Empires III, Counter-Strike: Source, Halo 2, Halo 3, Medal of Honor, Spider- Man 2, StarCraft II, The Matrix: Path of Neo, Tom Clancy's Splinter Cell: Chaos Theory.

Entre las posibilidades que ofrece Havok, las más significativas son:

- Primero se define el objeto **q** de tipo hkQuaternion
- Físicas rag-doll. Retomar el control del personaje una vez aplicadas las implicaciones físicas, de una manera más o menos realista. (Que para el caso de esta investigación posiblemente sea de poca utilidad)
- Sistema de físicas continúas. Permite que el motor físico actúe de forma permanente, sin estar sujeto a eventos predeterminados.(sumamente útil para cuando el aprendizaje por imitación requiere interacción con otros objetos)
- Portabilidad y facilidad de uso y, debido a estar basado en C/C++. Buen rendimiento debido a bajo consumo en memoria.

Dentro de las razones de elección para esta herramienta, son dos particularmente:

- La eliminación de tiempo a invertir en programar el entorno tridimensional con las implicaciones físicas que existen en el mundo real, pues Havok se

encarga de detectar colisiones, movimientos resultantes, efectos de la gravedad, etc.

- La presencia tan fuerte de la herramienta a nivel industrial y comercial, lo cual lo hace una herramienta sumamente confiable y de alta calidad.

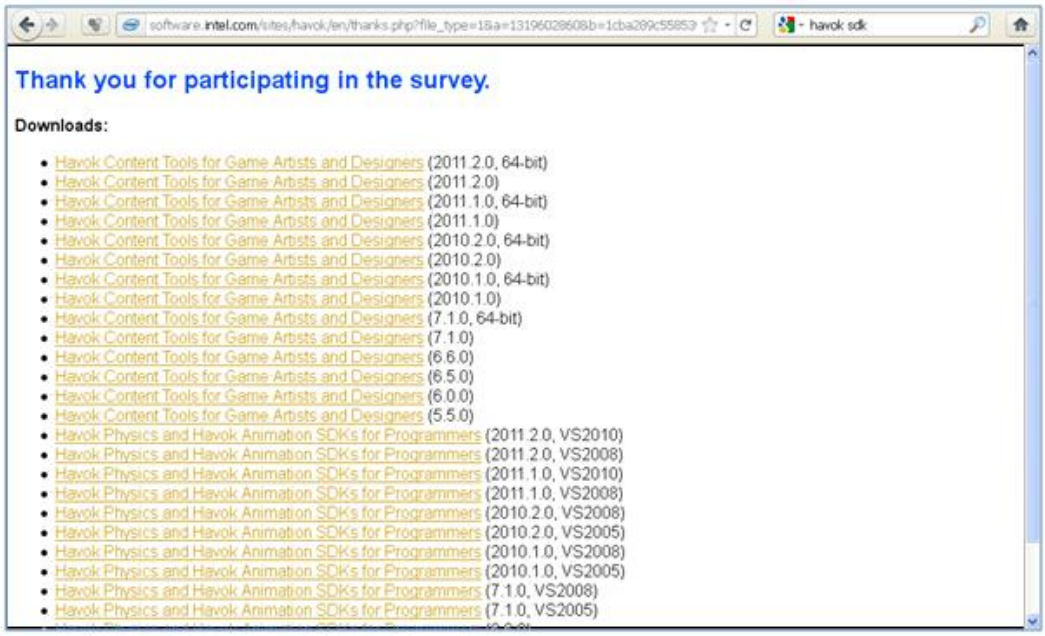
Software necesario para trabajar con Havok

Como parte del software que se necesita para trabajar con Havok se encuentra lo siguiente:

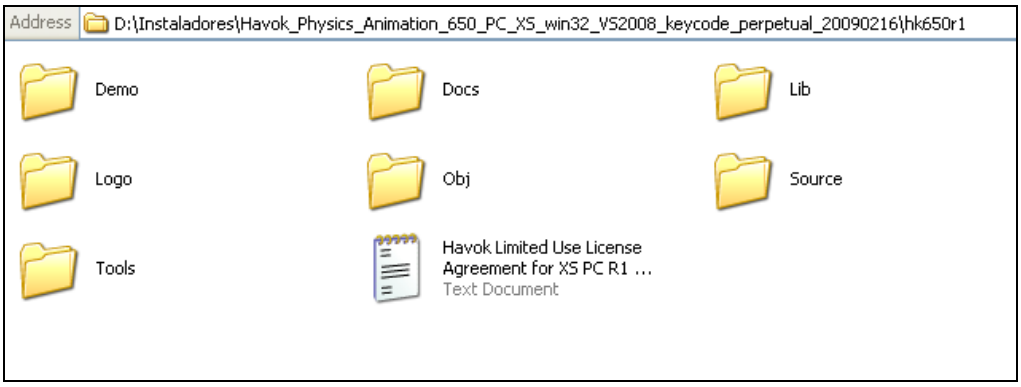
	Herramienta	Descripción
	Havok Behavior Tool for Game Artists and Designers (2011.2.0)	Este es un sistema para el desarrollo de eventos impulsados por los comportamientos de los personajes en un juego
	Havok Content Tools for Game Artists and Designers (2011.2.0)	Proporcionan una amplia y extensible conjunto de herramientas para la creación de activos, la exportación, transformación y carga
	Havok Physics and Havok Animation SDKs for Programmers (2011.2.0)	Este es el corazón de Havok , es el kit principal de herramientas de desarrollo, código fuente, archivos de cabecera, librerías, y archivos de implementación en c++ para lograr simulaciones rápidas en tiempo real enfocado en cuerpos rígidos.
	Visual C++ 2010 Express Edition	La herramienta recomendad como IDE y compilador para Havok.

	DirectX SDK	DirectX SDK es una excelente utilidad que ofrece un conjunto de aplicaciones destinadas al desarrollo de juegos y utilidades multimedia gestado con bases en DirectX. Dicha herramienta posee todo lo necesario para trabajar en lenguajes de programación tales como C#, C++, C, VisualBasic.NET que son en la actualidad, algunos de los lenguajes más utilizados y maleables, con los cuales será posible desarrollar todo tipo de juegos y utilidades multimedia
	Una herramienta de modelado para las cuales exista un plugin de exportación de archivos *.hkx, para las cuales están: • 3DS Max Tools • Maya Tools • XSI Tools	Con estas herramientas es posible crear modelos para ser importados dentro de código c++. Estas herramientas no son obligatorias pues se puede trabajar con los modelos de ejemplo de Havok, pero sin ellas no es posible crear nuevos modelos e importarlos a la animación.

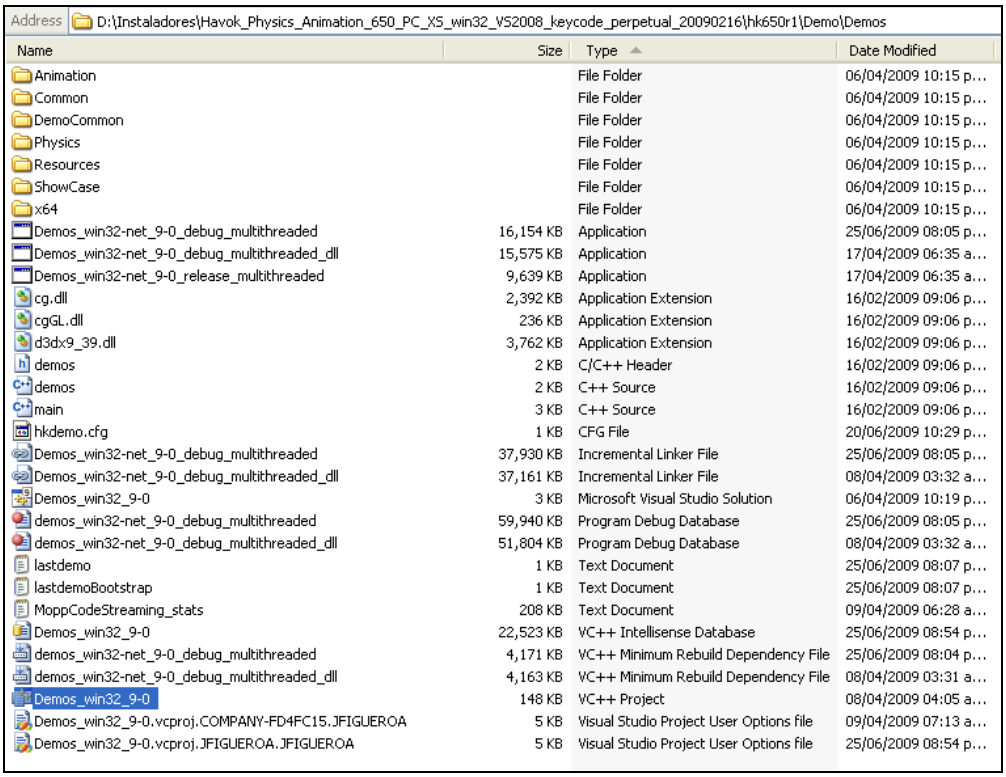
Las herramientas necesarias que conforman a Havok pueden descargarse de la siguiente dirección: http://software.intel.com/sites/havok/, donde después de registrar algunos datos como nombre, correo electrónico y motivo de la descarga, además de leer el contrato de licencia que especifica permitir el uso de Havok para fines no comerciales ya es posible acceder a los archivos.



Havok SDK proporciona todo lo necesario en cuanto a código fuente y herramientas para el desarrollo de programas en un mundo virtual, donde cabe destacar los múltiples ejemplos que brinda para su aprendizaje más a fondo, así como la documentación respectiva.



El conjunto de ejemplos que proporciona Havok SDK se encuentra en la carpeta Demos como se ve en la siguiente imagen.



Al momento de poner en ejecución dicho software en Visual C++ es posible apreciar un menú para visualizar cada ejemplo respectivamente, sin embargo, si antes no se ha leído detenidamente la documentación, entender el código resulta ser bastante complicado.