Análisis, diseño e implementación de un simulador de combate aéreo basado en lógica difusa

Francisco Cano [1], Fernando Juárez [1], Servando Ortega [1], Edmundo Camarillo [1], Leandro Balladares Ocaña [2]

[1] Escuela Superior de Computo del Instituto Politécnico Nacional, Unidad Profesional "Adolfo López Mateos" Zacatenco, C.P. 07738, México, México
duran_francisco@hotmail.com, son_fernando@hotmail.com, sislas2012@hotmail.com, edmundordc@hotmail.com

[2] Centro de Investigación en Computación del Instituto Politécnico Nacional, Unidad Profesional "Adolfo López Mateos" Zacatenco, C.P. 07738, México, México
ballad@cic.ipn.mx

Resumen. En este trabajo se describe el análisis, diseño y la implementación de un sistema de software de simulación de combate aéreo entre dos aeronave, llamado JIBA3D. Las aeronaves son controladas por el usuario y la computadora, respectivamente. El combate se lleva a cabo dentro de un espacio o mundo virtual tridimensional (3D) de dimensiones finitas, en el que hay obstáculos que las naves deben evitar. Se describe el análisis, diseño e implementación de cuatro módulos basados en lógica difusa, utilizados para controlar automáticamente la aeronave de la computadora, a saber: control de navegación, control de evasión / ataque, control de sistema de mira y control para proyectiles teledirigidos. El simulador opera y soporta interacción en tiempo real. La implementación fue realizada siguiendo la metodología de la programación orientada a objetos en el lenguaje C++.

1 Introducción

Actualmente, es innegable que una de las principales áreas de aplicación de las Ciencias Computacionales son los juegos interactivos basados en computadora. Dos de las áreas de las que más se han beneficiado los videojuegos actuales son: la graficación por computadora (en especial la graficación 3D en tiempo real) y la inteligencia artificial (IA). Tanto la IA, como las graficas 3D, tienen funciones importantes dentro del área computacional. La primera, como técnica para solucionar problemas en los que tiene que ver la experiencia humana y la segunda dentro del diseño y modelado de sistemas reales como virtuales [2–5], [7–12].

En la actualidad, existen un sinnúmero de sistemas de entretenimiento que hacen uso de graficas tridimensionales (3D), tales sistemas van desde juegos relativamente mples como el "Block Out" [5], hasta avanzados juegos de realidad virtual que hacen uso de dispositivos complejos como cascos, guantes para navegación o lentes

A. Gelbukh, M. Hernández Cruz (Eds.) Avances en la ciencia de la computación en México, CORE-2003, pp. 89–106, 2003.

para proyección estereoscópica [11], [12], [14]. Poro otro lado, la IA permite crear personajes o entidades que den la impresión de tener identidad, comportamiento, "inteligencia", haciendo que los juegos sean más reales, atractivos y presenten yor reto para los usuarios [9]. Para nuestro propósito, la conjunción de las gráficas 3D, la IA y técnicas de interacción, dará como resultado un sistema que involucra tanto la simulación del proceso de toma de decisiones humano simulado con lógica difusa [6], [13], como aspectos de graficación, como son la animación y la detección de colisiones en tiempo real. El desarrollo de este trabajo permitió comprobar utilidad y el buen desempeño de los sistemas difusos en la toma de decisiones ambientes dinámicos. El trabajo aquí presentado bien puede servir de base para desarrollo de simuladores de combate profesionales o para el desarrollo de sistemas de entretenimiento (videojuegos).

2 JIBA 3D

JIBA 3D es un sistema de simulación de combate aéreo basado en software entre dos naves dentro de un ambiente 3D. El juego consta de cinco tipos de naves a gir por el usuario y las naves restantes serán operadas por la computadora. El tipo batallas serán uno a uno, es decir cada quien sólo dispondrá de una nave (usuario y computadora). La vista que tendrá el usuario durante el juego es de tercera persona ya que este tipo de vista tiene un rango más amplio de visión. El ganador será aquel que logre eliminar a la otra nave primero, y ello lo conseguirá por medio del armamento con el que cuentan las naves, a saber: *rayos láser*, cuya trayectoria está definida por una línea recta que se prolonga hacia el punto donde apuntaba la mira de nave; *mísiles*, que pueden ser de *dirección fija* (son los mísiles que van en línea recta hacia el punto donde apuntaba la mira de la nave al momento de dispararlos) y *"teledirigidos"* (son aquellos mísiles que al ser disparados rastrean al enemigo dentro del campo de visión del misil y lo persiguen para tratar de hacer contacto.

Además, cada nave cuenta con un sistema de mira automática que permite ubicar al enemigo y dispararle con precisión, siempre y cuando se encuentre en el rango su campo de visión. Las naves se desplazan por un escenario de dimensiones finitas en forma de cubo. El usuario controla a su nave mediante entradas de teclado o *joystick*, mientras que la computadora, utiliza un sistema de toma de decisiones basado en lógica difusa.

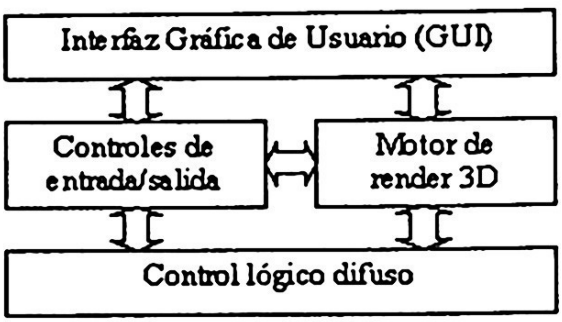


Fig. 1. Diagrama general de JIBA 3D.

En la Fig. 1 se presenta la arquitectura general de JIBA3D. En las siguientes secciones se describe brevemente cada uno de los módulos que la integran.

3 Modelado del sistema

La lógica difusa es una técnica que permite modelar sistemas expertos difusos, los cuales dan grandes ventajas sobre los sistemas expertos comunes. Una de las ventajas más importantes de tales sistemas es que permiten emplear el *razonamiento* aproximado, lo que hace al sistema más robusto y le da capacidad de tomar decisiones de forma muy parecida a como lo hace el hombre [6], [13]. Para desarrollar JIBA3D, se diseñaron e implementaron cuatro sistemas de control difuso:

- *sistema de navegación*,
- *sistema de evasion / ataque*,
- *sistema asistente de mira*, y
- *sistema guía de mísiles*.

Los pasos a seguir para el modelado de los sistemas fueron: 1) Establecer el conjunto de variables de entrada / salida. 2) Determinar los universos de discurso de cada una de estas variables y los conjuntos difusos para las mismas. En JIBA. 3D, se eligieron universos de discurso para la posiciones definidos en unidades de distancia y las direcciones en porcentaje de desplazamiento. La elección de porcentajes de desplazamiento, es debido a que estos podrán determinar más concretamente la razón dirección-velocidad con un patrón máximo de 1 adicional al estándar. 3) Definir las reglas, que determinarán el comportamiento del "ente-máquina" y que será esencial en su capacidad de respuesta. Las reglas, modelan la máquina de inferencia del sistema experto y se definen como del tipo `si ... entonces`. Enseguida, se presentan los sistemas de control modelados como casos aislados y se desarrolla en cada uno de ellos los tres pasos anteriores

3.1 Obtención de las entradas

Las entradas a los sistemas representan las posiciones relativas de los objetos del ambiente o de la nave del usuario, en cuyo caso también se incluye su vector de desplazamiento, y en base a las cuales cada uno de los sistemas difusos generará una acción. Ahora, se explicará como se obtiene cada una de ellas, ya que los parámetros que conocemos corresponden a valores del mundo en el que se desenvuelven y como tales proveen información útil para el buen funcionamiento de los sistemas difusos (ver Fig. 2a).

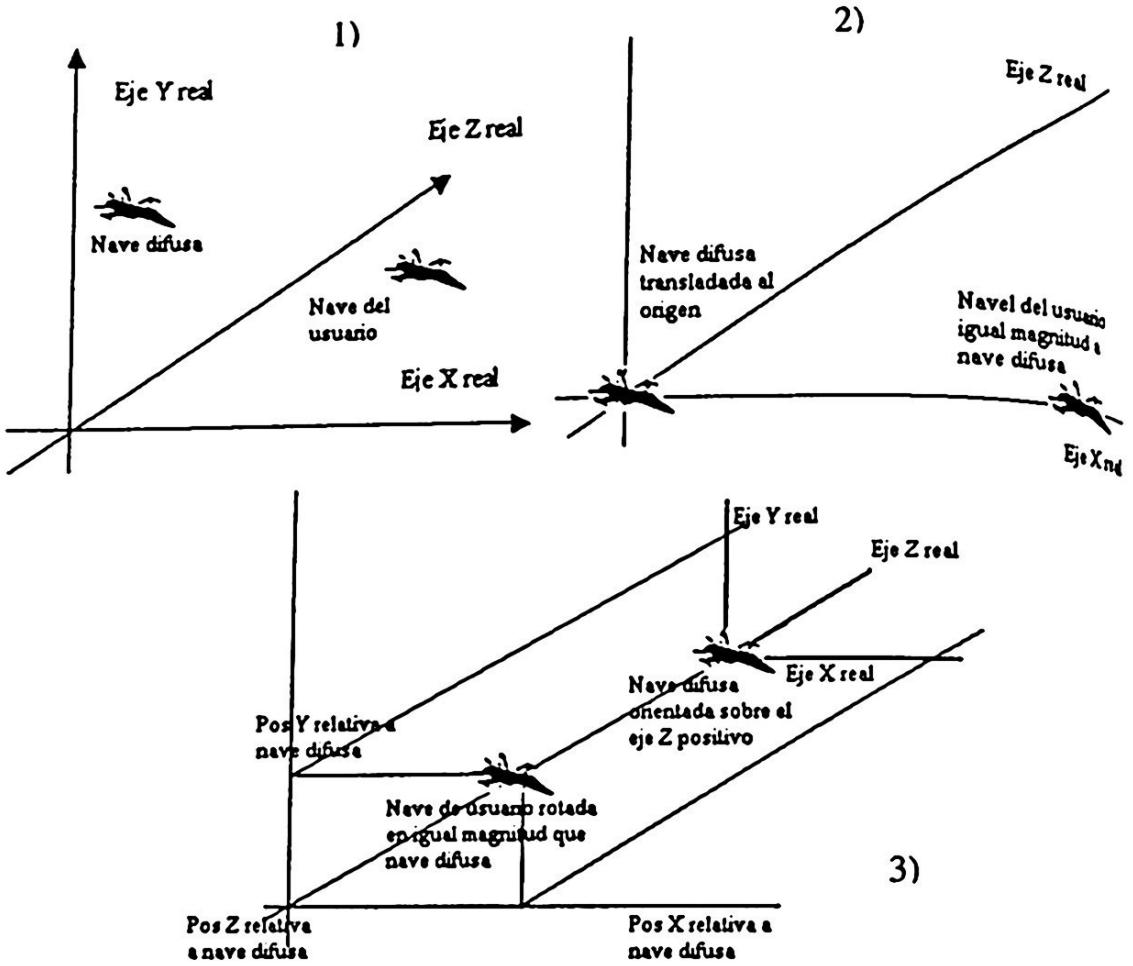


Fig. 2. 1) Posiciones iniciales de la nave difusa y del usuario en el mundo virtual, 2), 3) Transformaciones aplicadas a las posiciones de la nave difusa y del usuario en el mundo virtual.

El primer paso, antes de poder usar estos valores para que los sistemas difusos puedan interpretarlos de manera correcta, es llevar el espacio de coordenadas "universales", a coordenadas relativas a la nave difusa o al misil según sea el caso. Para llevar a cabo esa tarea, primero se debe trasladar a la nave o misil difusos, al "origen" del mundo, y trasladar, en la misma magnitud al objeto que se va a someter consideración por los sistemas difusos. Con esto, solo podemos conocer de manera aproximada, la posición relativa que guarda un elemento del escenario, o el usuario respecto a la nave o misil difusos (ver Fig. 2b).

El segundo paso, requiere hacer una transformación de rotación de cada una las componentes de posición relativa de la nave del usuario o del elemento del escenario, a fin de conocer la orientación de la posición de los mismos. Esta transformación es esencial, ya que todos los elementos del escenario mantienen una orientación "universal", la cual, dependiendo del tipo del elemento puede cambiar (tal el caso de las naves, ya sea la difusa o la del usuario, y la de los mísiles). Así pues la orientación de uno respecto a otro no es la misma en todo momento, y es necesario conocerla para determinar si estos se acercan, alejan o conservan la misma distancia (ver Fig. 2c).

El tercer paso, es interpretar cada uno de los valores obtenidos para que el sistema difuso pueda usarlos de manera correcta y este pueda generar una respuesta

satisfactoria. La interpretación de cada uno de los valores, se basó en la representación de un mapa cartesiano, así pues, para valores del eje de las X relativo, si estos son positivos, su significado semántico, es el de una posición relativa hacia la derecha del vector de desplazamiento de la nave o el misil difusos y un valor negativo significa una posición relativa hacia la izquierda del vector de desplazamiento; para el eje de la Y, un valor positivo significa una posición por sobre el vector de desplazamiento y un valor negativo una posición por debajo del vector de desplazamiento; y para el eje de la Z, un valor positivo significa una posición al frente del vector de desplazamiento y un valor negativo una posición detrás del mismo.

El cuarto paso, consiste en validar los valores obtenidos para determinar que control difuso es el que va a actuar. Para ese propósito, y en base a los universos de discurso de las variables de cada sistema, se someten a consenso en una primera instancia los elementos fijos del "mundo", esto es, todos aquellos elementos que podrían representar un obstáculo dentro del entorno virtual. Si alguno de ellos, representa en forma potencial un obstáculo dentro de nuestro espacio inmediato de desplazamiento, se pondrá en marcha el sistema difuso de navegación, el cual devolverá como respuesta componentes de desplazamiento que permitan a la nave difusa librar de la mejor manera dicho obstáculo. Si por el contrario, ningún elemento fijo cae en la situación anterior, el sistema difuso que entrará en funcionamiento será el de evasión ataque, que como se dijo, tiene la capacidad de generar una respuesta que permita a la nave difusa ganar ventaja sobre la nave del usuario.

3.2 Uso de variables de entrada en los sistemas de navegación – evasión / ataque

Modelado del sistema de navegación

El sistema de navegación, tiene como función librar cualquier obstáculo que se presente enfrente de la nave difusa, para ello, toma las entradas del sistema y efectúa una evaluación de ellas para determinar la posición relativa del obstáculo respecto de la nave. Si el obstáculo se encuentra en un rango dentro del espacio de desplazamiento inmediato de la nave difusa, se procede a hacer una "medición" de éste, tomando como datos las posiciones relativas de sus vértices, ya que todos los objetos inanimados del escenario son poligonales. Para obtener de manera adecuada esta información, se hace uso de tres variables temporales, las cuales van a almacenar las posiciones relativas de los vértices del objeto que se encuentran más alejadas la izquierda, a la derecha y hacia arriba respectivamente, ya que todos los objetos son cerrados y están fijos al piso del ambiente virtual. Habiendo realizado la medición del objeto, se someten a consenso las tres variables auxiliares, para obtener la de menor magnitud, ya que ésta nos permitirá encontrar la trayectoria menos forzada para evadir al obstáculo. Habiendo realizado el proceso anterior, el valor obtenido se envía como entrada al sistema difuso y este genera como salida una componente de desplazamiento ya sea para X, o Y, que junto con las otras dos, nos darán un desplazamiento real dentro de nuestro mundo virtual (ver Fig. 3). En el Anexo A.1 se lista la definición de las variables de entrada y matrices de reglas del sistema de navegación.

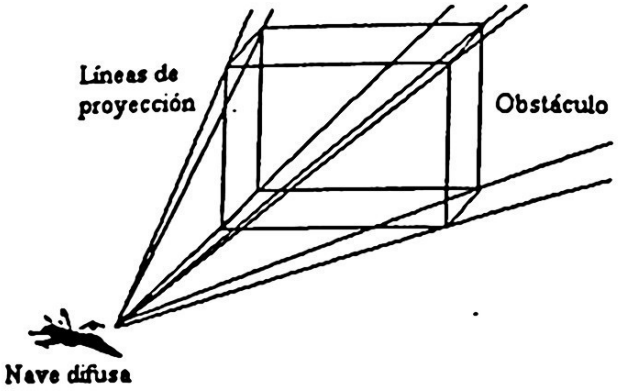


Fig. 3. Detección de obstáculos.

Modelado del sistema de evasión / ataque

En este sistema se detectaron dos variables generales de entrada y una de salida, las cuales se parten en tres cada una. La primera describe la posición relativa del objeto controlado por el usuario respecto del controlado por la máquina. La segunda, describe la dirección que el objeto controlado por el usuario, toma respecto del controlado por la máquina. La tercer variable, expresa o representa la dirección que debe tomar el objeto controlado por la máquina para obtener una ventaja sobre el usuario.

La primer variable, correspondiente a la posición relativa se divide en componentes relativas para los ejes X, Y, y Z respectivamente, y tomando siempre como origen a la nave difusa con dirección sobre el eje Z positivo. En el Anexo A.2, se muestran la definición del universo de discurso. La segunda variable, que describe la dirección relativa, se divide en componentes simples de desplazamiento relativo sobre los ejes X, Y, y Z como en el caso anterior, y que permitirán al control difuso "saber" como se comporta el usuario y que le permitirá definir su respuesta, ya sea de evasión o de ataque. En el Anexo A.2, se muestran las definiciones de sus universos de discurso. La tercer variable, que describe las variables de desplazamiento del objeto controlado por la máquina, se divide al igual que la segunda en componentes de desplazamiento relativo para cada uno de los ejes, y se obtendrán como salida del proceso de inferencia difuso del sistema de control. Los valores reales de estas variables, definirán el comportamiento de evasión o ataque de la nave difusa, por lo que la correcta interpretación de las mismas se traducirá en un desempeño óptimo de la nave difusa. En el Anexo A.2 se muestra la definición de ésta.

Habiendo definido los universos de discurso y los conjuntos, la tarea más importante es ahora modelar las reglas difusas y la máquina de inferencia. La definición de las reglas difusas, se hizo empleando el modelo difuso del sistema que se quería controlar y en base a un análisis del comportamiento del mismo. En el Anexo A.2 se muestran solo los casos generales en forma de matriz de reglas que se definieron para el modelado del sistema de evasión / ataque. En estas matrices de reglas, se pueden observar algunas casillas en blanco, debido a que estas se consideraron cruciales para la toma de decisiones y para la que se definió alrededor de 750 reglas específicas, que por cuestión de formato, no se incluyen en el artículo

El sistema de evasión / ataque, basa su funcionamiento en la posición relativa que el usuario representa respecto a la nave difusa, y dependiendo de esta, se considerará

o no el parámetro de desplazamiento relativo, que determinará si la acción a emprender es de evasión o ataque. Al validar la posición relativa del usuario respecto de la nave difusa, se puede determinar si esta se encuentra dentro de nuestro espacio de visión, es decir, si puede ser captada por nuestro "radar" virtual, con lo que se pondrá o no en marcha el sistema difuso. Si el usuario se encuentra visible a nuestro radar, entonces se tomarán sus parámetros de posición y desplazamiento relativos para enviarlos al control difuso. En caso contrario se considerará una visión parcial para cada una de las componentes, las cuales, en cada caso serán requeridas por la función difusa. Si ninguno de los casos anteriores se cumple, se efectuarán acciones que permitan a la nave difusa encaminarse al encuentro de la nave del usuario (ver Fig. 4).

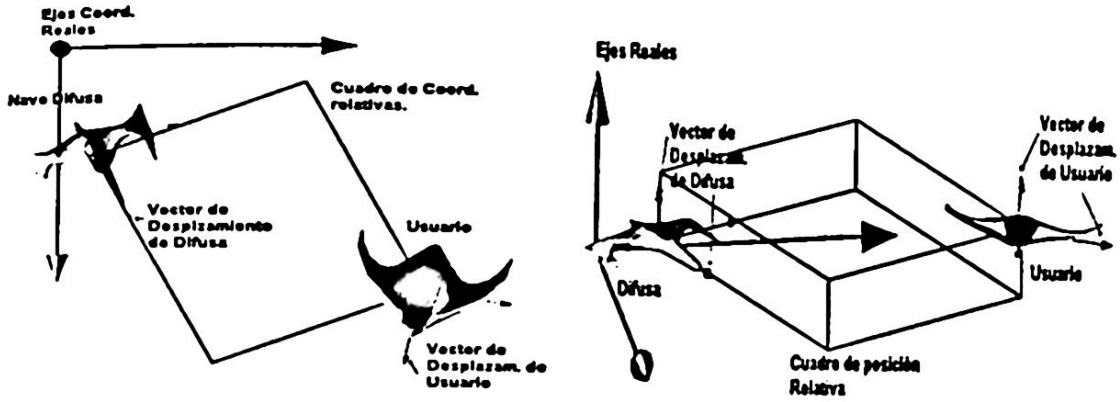


Fig. 4. Vista XZ y de perspectiva de situación inicial de evasión / ataque.

En cualquiera de los tres casos anteriores, la salida generada por el sistema simple o el difuso, será convertida a desplazamientos reales del mundo virtual que permitan a la nave difusa generar una respuesta que le beneficie, y permita obtener ventaja sobre el usuario (ver Fig. 5).

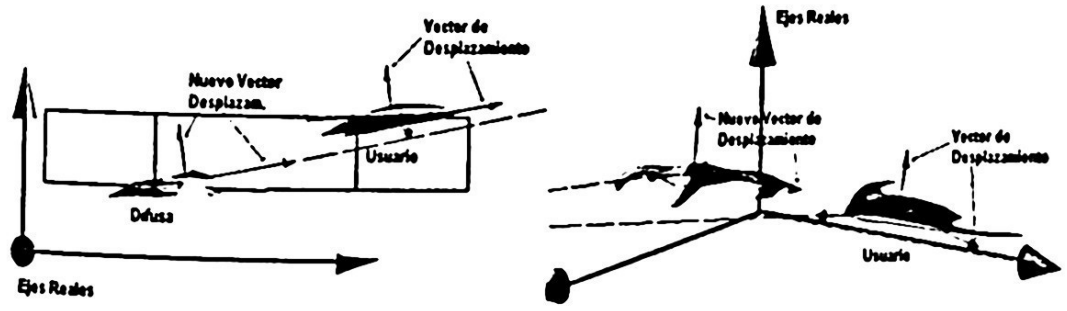


Fig. 5. Vistas auxiliares y de perspectiva de situación final de evasión / ataque.

Un caso especial se debe considerar cuando la nave difusa se acerca a los límites permitidos del mundo virtual, ya que no debe por ningún motivo abandonar al escenario, independientemente de donde se encuentre el usuario. En ese caso, se debe emprender una acción temporal que obligue a la nave difusa a regresar al espacio

permitido de acción. En un principio, se pensó en crear un sistema que controlara esa situación, pero con el modelado del sistema de evasión / ataque, se observó que se podía prescindir del mismo, pues el problema se solucionaba con hacer, por medio del sistema de evasión / ataque, que la nave difusa considerará un enemigo virtual situado en el centro del escenario. Así, en cualquier caso en que la nave difusa se acercara a los límites externos del mundo virtual, emprendería por un tiempo muy limitado un ataque contra ese enemigo virtual simulando el regreso al escenario eliminando así la posibilidad de dejar al usuario sin enemigo.

La puesta en funcionamiento de uno u otro sistema (navegación o evasión / ataque), dependerá de la situación que se presente, así pues, como se explicó en último punto de la sección anterior, si se tiene un obstáculo dentro del espacio inmediato de desplazamiento, el sistema que se pondrá en marcha será el de navegación.

3.3 Uso de variables de entrada en los sistemas de asistente de mira y guía de misiles

Los sistemas de asistente de mira y de guía de misiles son muy parecidos en cuanto a su funcionamiento, ya que ambos tienen una región ciega en el espacio inmediato de desplazamiento, que permite al usuario y a la nave difusa según sea el caso, evitar ser "detectados" por el mecanismo estricto de ataque, es decir, por la ejecución de una acción de disparo de misiles difusos, implícita por la detección del objetivo en la mira. En el espacio inmediato a esta región ciega, se encuentra una pirámide que representa el espacio de visión de dichos sistemas. Como dicho espacio no uniforme, debido a su forma piramidal creciente, se detectó la ventaja de poder aumentar la precisión en la localización de las naves, y en la estimación de su posición siguiente inmediata por lo que se empleó un factor de escalamiento que permite hacer dicha transformación independientemente de la posición sobre el eje de desplazamiento inmediato relativo de la nave que haga uso de dicho sistema.

El sistema de guía de misiles, emplea una representación bidimensional de la posición y desplazamiento relativos del objetivo para determinar si tiene que cambiar o no sus componentes de desplazamiento, situarse sobre el eje de desplazamiento inmediato y poder hacer contacto con este. Por lo tanto, la salida generada está compuesta por un vector tridimensional de desplazamiento real que define la trayectoria inmediata del misil.

Sistema asistente de mira

Para que el usuario no se preocupe por apuntar hacia las naves enemigas, se diseñó una mira "inteligente" capaz de seguir, en un cierto rango de visión, a la nave enemiga más cercana. Para ello se definieron los conjuntos de entrada y salida respectivos, así como sus universos de discurso, conjuntos difusos, tipos de funciones membresía y reglas difusas (ver Anexo A.3).

Conjuntos de entrada y salida

Para poder conocer como se va a desplazar la mira es necesario conocer la posición de la nave enemiga y su desplazamiento. Lo anterior permitió concluir que se necesitaban los siguientes conjuntos de entrada: posición en X, Y, Z de la nave enemiga, así como su desplazamiento sobre cada eje. Como salida era necesario saber la dirección sobre los ejes X, Y, Z en la que se desplazará la mira, es decir, sus nuevos desplazamientos.

Definición de conjuntos difusos

Antes de poder definir los conjuntos difusos de las variables de entrada y salida, es necesario definir primero el universo de discurso de las variables, así como la forma en que la mira va a ver al enemigo. Primero definiremos como es el campo de visión de la mira o región de visión. La mira va a poder localizar al enemigo siempre y cuando este se localice en el campo de visión especificado para la mira. Para nuestro caso el campo de visión será una pirámide. Se escoge la pirámide porque es muy similar a la vista que se tendría en la realidad (entre más cerca menos se ve, o menor rango, y entre más lejos es mayor la visión). La punta de la pirámide estará en las coordenadas de la nave, pero, como es de suponer, la mira no podrá desplazarse hasta este punto, ya que esto querría decir que: (1) podemos dejar que el enemigo se acerque demasiado; (2) el espacio con el que se trabaja es del tamaño de un punto (ver Fig. 6).

En vez de eso, consideraremos que la mira comienza a funcionar en una parte delante de la nave. Otra consideración que hay que realizar antes de comenzar, es que la mira tomará como origen a la nave en la que está implementada, así como las coordenadas relativas con respecto a ella de la nave enemiga.

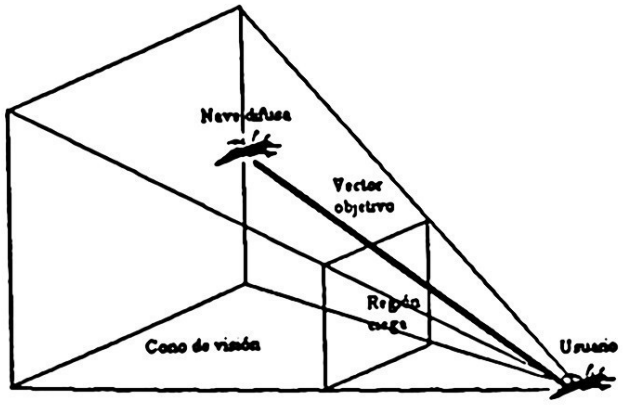


Fig. 6. Definición de la pirámide de visión de las naves.

Conjuntos difusos de entrada

Con esto podemos empezar a definir el universo de discurso. Para `XPosU` y `YPosU` tendremos conjuntos idénticos (sólo cambiarán los nombres de la etiquetas) así como universos de discurso, ya que como se puede apreciar en el Anexo A.3 siempre se tendrán cuadros en lo que a X y Y se refiere. En el caso del eje Z no es igual, ya que en este no podemos considerar posiciones negativas, sino que, es desde una

posición delante del origen hasta otra. Los desplazamientos de XDesU y YDesU como en los casos de las posiciones, son iguales (ver Anexo A.3).

Conjuntos difusos de salida
Los conjuntos de salida quedan de la forma mostrada en el Anexo A, que es idéntica a los desplazamientos en los ejes.

Definición de la base de reglas difusas
Una vez definidos los conjuntos de entrada y salida, sus universos de discurso, menzaremos a realizar la base de reglas para la mira inteligente. Para ello realizaremos módulos que controlen el movimiento en cada eje y simplificar así la base reglas. Comenzaremos con el movimiento sobre el eje X (cabe señalar que iguales los conjuntos X y Y, su base de reglas es idéntica solo cambiando las etiquetas correspondientes). Ver las reglas en el Anexo A.

Sistema de guía de mísiles
Cuando cualquiera de las naves (del usuario y difusa) disparen mísiles, tendrán posibilidad de que sean de dos tipos: misil directo y misil guiado. El misil directo tiene como característica que siempre va en una sola dirección y nunca cambia; cambio los mísiles guiados cuentan con un sistema difuso que les permite local la nave enemiga e ir directo hacia ella para hacer impacto. En esta sección se hablará del sistema difuso que se modeló para ello.

Conjuntos de entrada y salida
Para poder conocer como se va a desplazar el mira necesitamos conocer la posición de la nave enemiga y su desplazamiento. Lo anterior nos lleva a concluir que necesitamos los siguientes conjuntos de entrada: posición en X, Y, Z de la nave enemiga (XPosU, YPosU, ZposU respectivamente), así como su desplazamiento sobre eje (XDesU, YDesU y ZDesU). Como salida necesitamos saber la dirección los ejes X, Y, Z en la que se desplazará el misil, es decir, sus nuevos desplazamientos: XNDes, y YNDes (el desplazamiento sobre el eje Z no se considera porque siempre es hacia delante y constante)

Definición de conjuntos difusos
Los conjunto difusos utilizados para el sistema difuso del misil, son los mismos en la mira, solo se elimina el sistema que controla el movimiento sobre el eje También cabe mencionar que el misil contará con una región ciega al igual que mira, solo que este es más amplio. Otra consideración que hay que realizar antes comenzar, es que el misil tomará como origen a la nave en de la que saldrá para implementada, y después el control difuso modificará sus coordenadas.

4 Motor de *rendering* 3D

El proceso de síntesis de la escena se dividió en dos procesos; el procesamiento geométrico y el proceso *raster*. En el primero de ellos se involucró a los procesos de transformación de coordenadas reales a coordenadas de visión, y fue donde se empleó los procesos de recorte o *clipping* y proyección de la escena. La Fig. 7 muestra los módulos de los que se compone el motor de *render* 3D de JIBA3D.

En JIBA3D se utilizó proyección de perspectiva, en la que se emplea un cono cuadrado de visión, que permite crear un efecto de profundidad en la imagen generada y da mayor realismo a la escena sintetizada, además de que este tipo de proyección puede ser realizada en tiempo real[1].

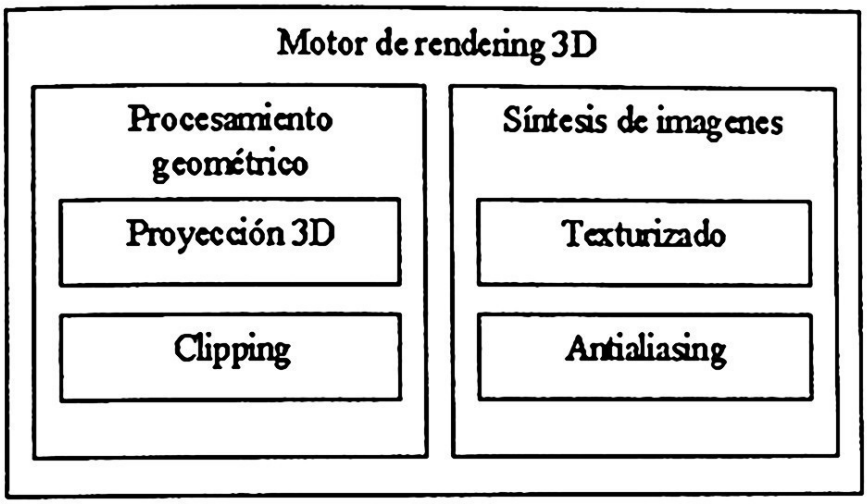


Fig. 7. Diagrama a bloques del motor de render 3D.

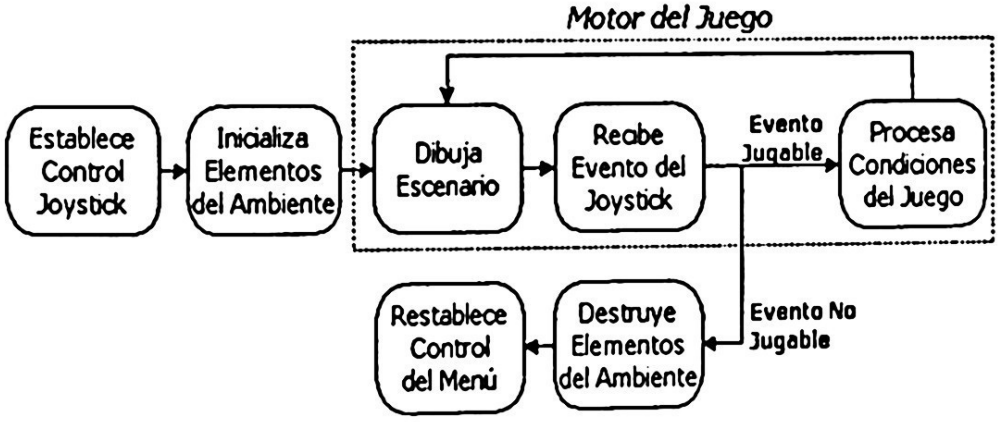


Fig. 8. Motor del juego.

[1] Existen otras técnicas que utilizan algoritmos de iluminación como el *ray tracing* - trazado de rayos - que permiten dar gran calidad a las imágenes de las escenas, pero son muy costosas en cuanto al tiempo de procesamiento y solo son usadas para crear animaciones que no requieren ser generadas en tiempo real.

5 Motor del juego

El motor del juego es el núcleo de JIBA3D y quien integra la funcionalidad de módulos descritos anteriormente para dar vida al sistema, además implementa lógica del mismo.

En la Fig. 8 se ilustra el contexto del motor de juego y su papel en JIBA3D.

6 Tecnologías de desarrollo

JIBA3D fue implementado sobre Windows 98 utilizando el lenguaje de programación C++. Se utilizó la herramienta Borland C++ Builder 5.0. El motor de render implementó utilizando el API GLIDE de la empresa 3dfx Interactive.

Para la implementación de los controles difusos, se utilizó la herramienta FIDE [1], que permite definir de manera muy sencilla y con reglas sintácticas simples un sistema difuso completo, además de generar el código fuente en ANSI C sistema modelado, en forma de una función que recibe como parámetros los valores de las variables de entrada sin fusificar y retorna los valores de las variables de salida defusificadas. El aspecto anterior dio gran apoyo a la tarea de diseño y parte de implementación de nuestros sistemas de toma de decisiones, ya que no fue necesario desarrollar los procesos de fusificación, la máquina de inferencia y defusificación. Un punto importante, es el hecho de que con esta herramienta se pudo observar aunque a un nivel muy básico el comportamiento de nuestros sistemas, lo que nos permitió hacer estimaciones de comportamiento en aplicaciones reales.

Referencias

1. Aptronix, Inc. (1994), "*FIDE: Fuzzy Inference Development Environment*".
2. Watt (1993), "*3D Computer Graphics*", Addison-Wesley.
3. Watt & F. Policarpio (2001), "*3D Games Real-time Rendering and Software Techonology*", Volume One, Addison-Wesley.
4. Watt, S. Maddock (2000), "*Computer Games Techonolgy and Higher Education*", Virtual Reality: Research, Development and Applications, Volume 5, Number 4, p. 185-194.
5. "*Block Out*", URL: http://www.xdgames.com/games/borx/
6. Chin-teng Lin & C.S. George Lee (1996), "*Neural Fuzzy Systems*", Prentince Hall.
7. Durán C. Fco, et al (2001), "*Manual de Referencia Técnica de JIBA3D*", Trabajo Terminal TTR0033 de la ESCOM -IPN.
8. John E. Laird (2002), "*Research in Human – Level AI Using Computer Games*", Communications of the ACM, January 2002, Vol. 45, Number 1, p. 32-35.
9. M. Cavazza (2000), "*AI in Computer Games: Survey and Perspectives*", Virtual Reality: Research, Development and Applications, Volume 5, Number 4, p. 223-235.

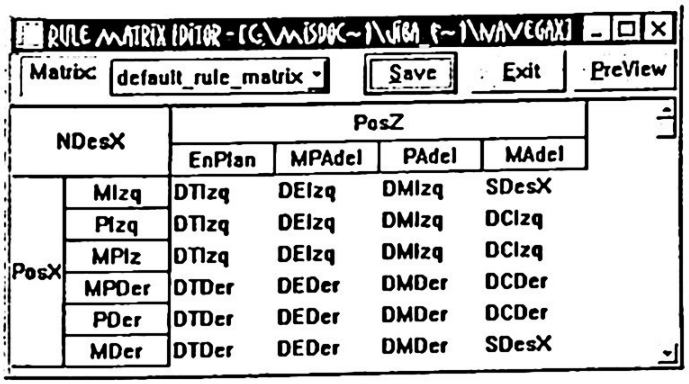
10. Michael Lewis & Jeffrey Jacobson (2002), "*Game Engines in Scientific Research*", Communications of the ACM, January 2002, Vol. 45, Number 1, p. 27-31.
11. Scientific American (2002), "*Information Techology: Augmented Reality: A New Way of Seeing*", April 2002, p. 34.
12. Wayne P. & Bruce T. (2002), "*ARQuake: The Outdoor Augmented Reality Gaming System*", Communications of the ACM, January 2002, Vol. 45, Number 1, p. 36-38.
13. Witold Pedrycz (1993), "*Fuzzy Control and Fuzzy Systems*", Research Studies Press.
14. Wolgang B., et al (2001), "*Projects in VR*", IEEE Computer Graphics and Applications, November/December 2001, p. 14-17.

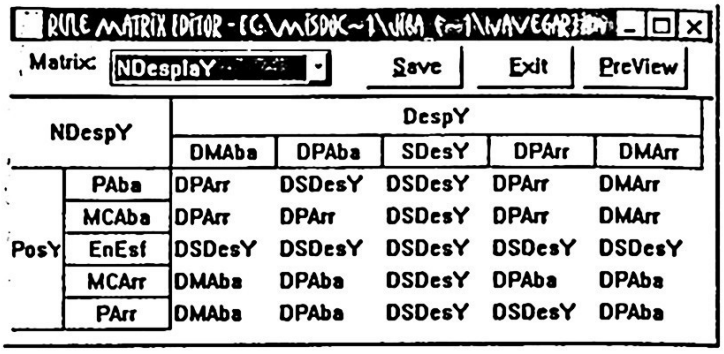
Anexo A – Modelado de los sistemas difusos

A.1 Variables de entrada y matrices de reglas del sistema de navegación

```
invar PosX "" :-20.0 (  ) 20.0 [
   MIzq (@-20.0, 1, @-15.0, 1, @-10.0, 0),
   PIzq (@-15.0, 0, @-10.0, 1, @-5.0, 0),
   CIzq (@-10.0, 0, @-5.0, 1, @0.0, 0),
   CDer (@0.0, 0, @5.0, 1, @10.0, 0),
   PDer (@5.0, 0, @10.0, 1, @15.0, 0),
   MDer (@10.0, 0, @15.0, 1, @20.0, 1)
];
invar PosZ "" :0.0 (  ) 30.0 [
   Colision(@0.0, 1, @5.0, 0),
   EnFrente(@0.0, 0, @5.0, 1, @10.0, 0),
   MCerca(@5.0, 0, @10.0, 1, @15.0, 0),
   PCerca(@10.0, 0, @16.0, 1, @22.0, 0),
   Normal (@16.0, 0, @22.0, 1, @30.0, 1)
];
invar PosY "" :0.0 (  ) 25.0 [
   Arriba(@0.0, 0, @10.0, 1, @15.0, 1),
   PArr (@10.0, 0, @15.0, 1, @20.0, 0),
   MArr (@15.0, 0, @20.0, 1, @25.0, 1 )
];
```

donde: `PosX`, `PosY` y `PosZ` son las componentes de posición relativas del objeto ambiental respecto a la nave difusa, Las palabras al inicio de cada línea son las etiquetas de los conjuntos difusos, los valores precedidos por arriba definen los rangos de cada conjunto difuso, seguido de su valor de pertenencia.





A.2 Definición de variables de entrada de posición y desplazamiento del sistema de evasión / ataque

```
invar PosZ "Pix" :-240.0 (  ) 240.0 [
   HAtras (@-240.0, 1, @-180.0, 1, @-120.0, 0),
   MuAtras (@-180.0, 0, @-120.0, 1, @-80.0, 0),
   MeAtras (@-120.0, 0, @-80.0, 1, @-40.0, 0),
   PAtras (@-80.0, 0, @-40.0, 1, @0.0, 0),
   EnPlan (@-10.0, 0, @0.0, 1, @10.0, 0),
   PAdel (@0.0, 0, @40.0, 1, @80.0, 0),
   MeAdel (@40.0, 0, @80.0, 1, @120.0, 0),
   MuAdel (@80.0, 0, @120.0, 1, @180.0, 0),
   HAdel (@120.0, 0, @180.0, 1, @240, 1)
];
invar PosZ "Pix" :-240.0 (  ) 240.0 [
   HAtras (@-240.0, 1, @-180.0, 1, @-120.0, 0),
   MuAtras (@-180.0, 0, @-120.0, 1, @-80.0, 0),
   MeAtras (@-120.0, 0, @-80.0, 1, @-40.0, 0),
   PAtras (@-80.0, 0, @-40.0, 1, @0.0, 0),
   EnPlan (@-10.0, 0, @0.0, 1, @10.0, 0),
   PAdel (@0.0, 0, @40.0, 1, @80.0, 0),
   MeAdel (@40.0, 0, @80.0, 1, @120.0, 0),
   MuAdel (@80.0, 0, @120.0, 1, @180.0, 0),
   HAdel (@120.0, 0, @180.0, 1, @240, 1)
];
invar PosY "Pix" :-240.0 (  ) 240.0 [
   HAba (@-240.0, 1, @-180.0, 1, @-120.0, 0),
   MuAba (@-180.0, 0, @-120.0, 1, @-80.0, 0),
   MeAba(@-120.0, 0, @-80.0, 1, @-40.0, 0),
   PAba (@-80.0, 0, @-40.0, 1, @0.0, 0),
   EnNiv (@-50.0, 0, @0.0, 1, @5.0, 0),
   PArr (@0.0, 0, @40.0, 1, @80.0, 0),
   MeArr (@40.0, 0, @80.0, 1, @120.0, 0),
   MuArr (@80.0, 0, @120.0, 1, @180.0, 0),
   Harr (@120.0, 0, @180.0, 1, @240.0, 1)
];
```

donde: PosX, PosY y PosZ representan las componentes de posición relativas del usuario respecto a la nave difusa, Las palabras al inicio de cada línea son las etiquetas de los conjuntos difusos, los valores precedidos por arriba definen los rangos de cada conjunto difuso seguido de su valor de pertenencia.

```
invar DesX "" :-1.000000 (  ) 1.000000 [
   DTIzq (@-1.0, 1, @-0.75, 0),
   DEIzq (@-1.0, 0, @-0.75, 1, @-0.5, 0),
   DMIzq (@-0.75, 0, @-0.5, 1, @-0.25, 0),
   DCIzq (@-0.5, 0, @-0.25, 1, @0.0, 0),
   SDesX (@-1.0, 0, @0.0, 0, @1.0, 0.0),
   DCDer (@0.0, 0, @0.25, 1, @0.5, 0),
   DMDer (@0.25, 0, @0.5, 1, @0.75, 0),
   DEDer (@0.5, 0, @0.75, 1, @1.0, 0),
   DTDer (@0.75, 0, @1.0, 1)
];
invar DesZ "" :-1.0 (  ) 1.000000 [
   DTAtras (@-1.0, 1, @-0.75, 1,@-0.55, 0),
   DEAtras  (@-0.75, 0, @-0.55, 1, @-0.35, 0),
   DMAtras (@-0.55, 0, @-0.35, 1, @-0.15, 0),
   DCAtras  (@-0.35, 0, @-0.15, 1, @0.0, 0),
   SDesZ (@-0.15, 0, @0.0, 1, @0.15, 0),
   DCAdel (@0.0, 0, @0.15, 1, @0.35, 0),
   DMAdel (@0.15, 0, @0.35, 1, @0.55, 0),
   DEAdel (@0.35, 0, @0.55, 1, @0.75, 0),
   DTAdel (@0.55, 0, @0.75, 1, @1.0, 1)
];
invar DesY "" :-1.0 (  ) 1.0 [
   DTAba (@-1.0, 1, @-0.75, 1,@-0.55, 0),
   DEAba (@-0.75, 0, @-0.55, 1, @-0.35, 0),
   DMAba (@-0.55, 0, @-0.35, 1, @-0.15, 0),
   DCAba (@-0.35, 0, @-0.15, 1, @0.0, 0),
   SDesY (@-0.15, 0, @0.0, 1, @0.15, 0),
   DCArr (@0.0, 0, @0.15, 1, @0.35, 0),
   DMArr (@0.15, 0, @0.35, 1, @0.55, 0),
   DEArr (@0.35, 0, @0.55, 1, @0.75, 0),
   DTArr (@0.55, 0, @0.75, 1, @1.0, 1)
];
```

donde: `DesX`, `DesY` y `DesZ` representan las componentes relativas de desplazamiento del usuario respecto a la nave difusa, Las palabras al inicio de cada línea son las etiquetas de los conjuntos difusos, los valores precedidos por arriba definen los rangos de cada conjunto difuso, seguido de su valor de pertenencia.

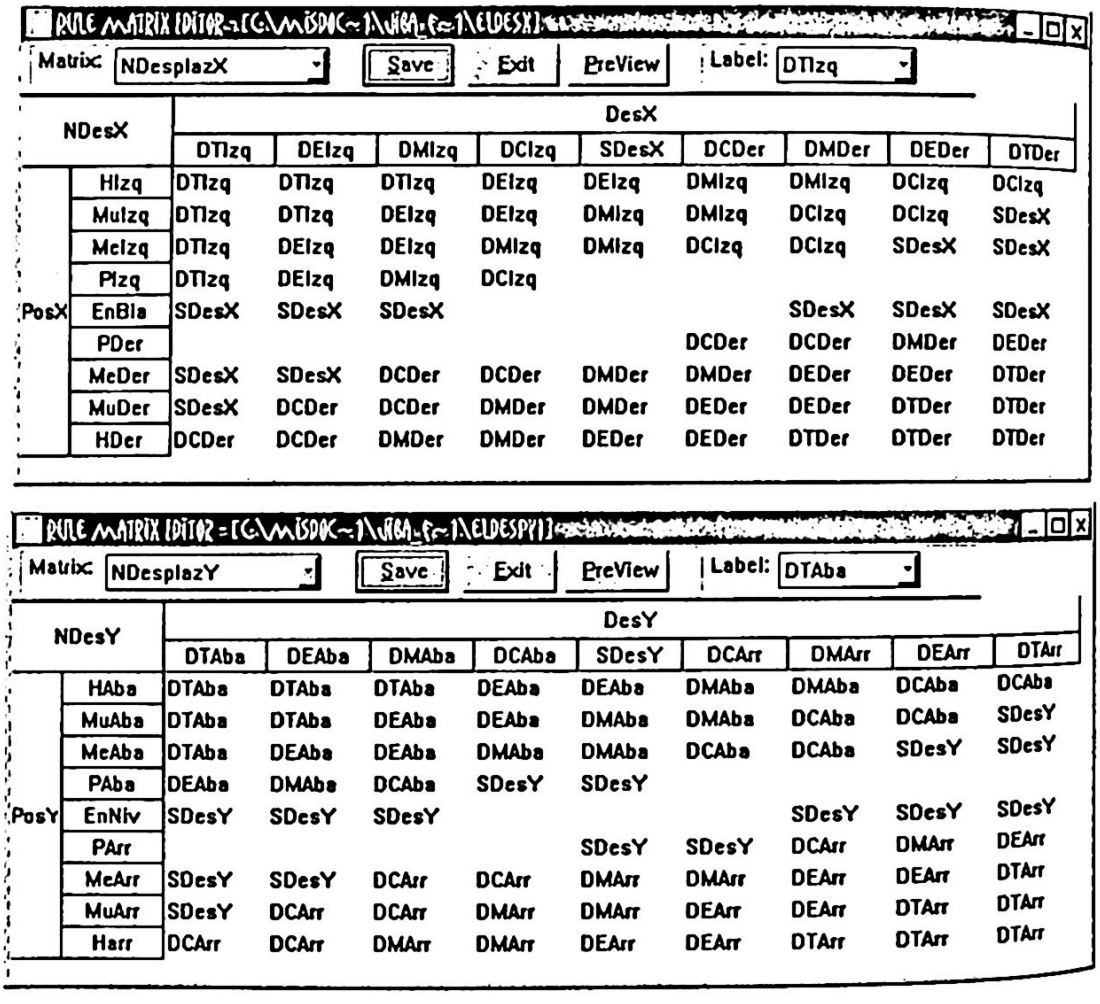
Variable de salida de desplazamiento del sistema de evasión / ataque

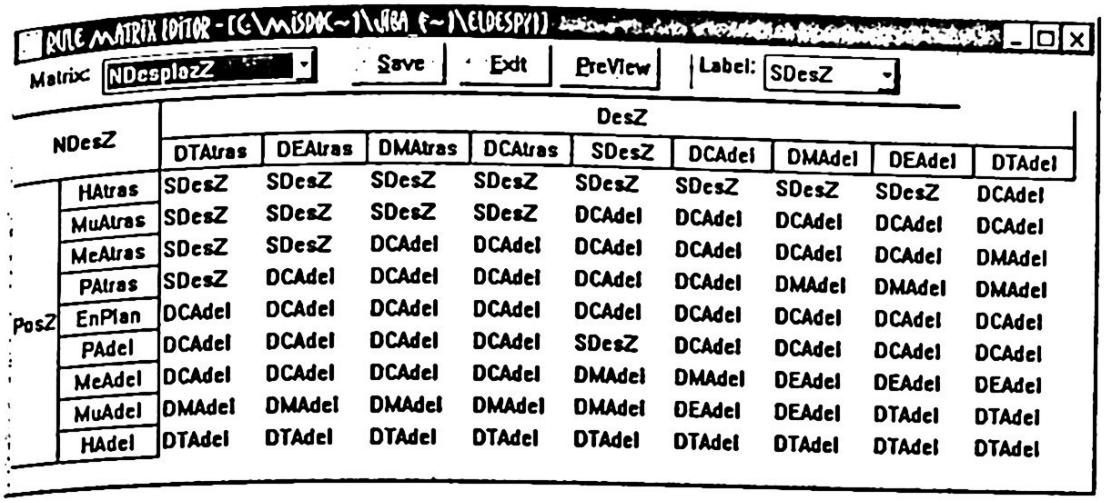
```
outvar NDesX "" :-0.25 ( ) 0.25 centroid [
   DTIzq (@-0.25, 1, @0.15, 0 ),
   DEIzq (@-0.20, 0, @-0.15, 1, @-0.10, 0),
   DMIzq (@-0.15, 0, @-0.10, 1, @-0.05, 0),
   DCIzq (@-0.10, 0, @-0.05, 1, @0.0, 0),
   SDesX (@-0.05, 0.00, @-0.0, 1, @0.05, 0),
   DCDer (@0.0, 0, @0.05, 1, @0.10, 0),
   DMDer (@0.05, 0, @0.10, 1, @0.15, 0),
   DEDer (@0.10, 0, @0.15, 1, @0.20, 0),
   DTDer (@0.15, 0, @0.25, 1)
];
outvar NDesZ "" :0.0 ( ) 1.0 centroid [
   SDesZ (@0.000000, 1, @0.25, 0),
   DCAdel (@0.0, 0, @0.25, 1, @0.50, 0),
```

```
    DMAdel (@0.25, 0, @0.50, 1, @0.750, 0),
    DEAdel (@0.50, 0, @0.75, 1, @1.0, 0),
    DTAdel (@0.75, 0, @1.0, 1)
]
outvar NDesY "" :-0.25 ( ) 0.25 centroid [
    DTAba (@-0.25, 1, @-0.20, 1, @-0.15, 0),
    DEAba (@-0.20, 0, @-0.15, 1, @-0.10, 0),
    DMAba (@-0.15, 0, @-0.10, 1, @-0.050, 0),
    DCAba (@-0.10, 0, @-0.050, 1, @0.0, 0),
    SDesY (@-0.05, 0, @0.0, 1, @0.05, 0),
    DCArr (@0.0, 0, @0.05, 1, @0.10, 0),
    DMArr (@0.05, 0, @0.10, 1, @0.15, 0),
    DEArr (@0.10, 0, @0.15, 1, @0.20, 0),
    DTArr (@0.15, 0, @0.20, 1, @0.25, 1)
];
```

donde: NDesX, NDesY y NDesZ representan las componentes relativas de desplazami que deberá tomar la nave difusa respecto del usuario, Las palabras al inicio de cada línea las etiquetas de los conjuntos difusos, los valores precedidos por arriba definen los rango cada conjunto difuso, seguido de su valor de pertenencia.

Matrices de reglas




RULE MATRIX EDITOR - [C:\MISDOC~1\JUEGA~1\ELDESPY1]

Matrix: NDesplozZ | Save | Exit | PreView | Label: SDesZ

NDesZ	DesZ								
	DTAtras	DEAtras	DMAtras	DCAtras	SDesZ	DCAdel	DMAdel	DEAdel	DTAdel
HAtras	SDesZ	SDesZ	SDesZ	SDesZ	SDesZ	SDesZ	SDesZ	SDesZ	DCAdel
MuAtras	SDesZ	SDesZ	SDesZ	SDesZ	DCAdel	DCAdel	DCAdel	DCAdel	DCAdel
MeAtras	SDesZ	SDesZ	DCAdel	DCAdel	DCAdel	DCAdel	DCAdel	DCAdel	DMAdel
PAtras	SDesZ	DCAdel	DCAdel	DCAdel	DCAdel	DCAdel	DMAdel	DMAdel	DMAdel
PosZ EnPlan	DCAdel	DCAdel	DCAdel	DCAdel	DCAdel	DCAdel	DCAdel	DCAdel	DCAdel
PAdel	DCAdel	DCAdel	DCAdel	DCAdel	SDesZ	DCAdel	DCAdel	DCAdel	DCAdel
MeAdel	DCAdel	DCAdel	DCAdel	DCAdel	DMAdel	DMAdel	DEAdel	DEAdel	DEAdel
MuAdel	DMAdel	DMAdel	DMAdel	DMAdel	DMAdel	DEAdel	DEAdel	DTAdel	DTAdel
HAdel	DTAdel	DTAdel	DTAdel	DTAdel	DTAdel	DTAdel	DTAdel	DTAdel	DTAdel

A.3. Sistema asistente de mira

Definición de variables de entrada de posición X, Y y Z

```
invar XPosU "pix" :-125 (1) 125 [
   DIzq (@-125, 1,@-91,1, @-62, 0),
   MIzq (@-91, 0, @-62, 1, @-31, 0),
   PIzq (@-50, 0, @-20, 1, @0, 0),
   EnplaX (@-20, 0, @0, 1, @20, 0),
   PDer (@0, 0, @20, 1, @50, 0),
   MDer (@31, 0, @62, 1, @91, 0),
   DDer (@62, 0,@91,1, @125, 1)
];
invar YPosU "pix" :-125 (1) 125 [
   DAba (@-125,1,@-91,1, @-50, 0),
   MAba (@-91, 0, @-62, 1, @-31, 0),
   PAba (@-62, 0, @-20, 1, @0, 0),
   EnplaY (@-20, 0, @0, 1, @20, 0),
   PArr (@0, 0, @20, 1, @50, 0),
   MArr (@31, 0, @62, 1, @91, 0),
   DArr (@62, 0,@91,1, @125, 1)
];
invar ZPosU "pix" :100 ( 1 ) 350 [
   Enfrente (@100, 1, @135, 0),
   MCerca (@100, 0, @135, 1, @170, 0),
   Cerca (@135, 0, @170, 1, @205, 0),
   MeCerca (@170, 0, @205, 1, @240, 0),
   MLejos (@205, 0, @240, 1, @275,0),
   Lejos(@240, 0, @275, 1, @310,0 ),
   MuLejos( @275, 0, @310, 1, @350,1)
];
```

Definición de variables de entrada de desplazamiento

```
invar XDesU "" :-1.000000 (  ) 1.000000 [
   Izq (@-1, 1, @-0.75, 1,@-0.5,0),
   PIzqu (@-0.75, 0, @-0.25, 1, @0, 0),
   SinDesX (@-0.25, 0, @0, 1, @0.25, 0),
   PDere (@0, 0, @0.25, 1, @0.75, 0),
   Der (@0.5, 0, @0.75, 1, @1,1)
];
```

```
invar YDesU "" :-1.000000 (  ) 1.000000 [
   Abajo (@-1, 1, @-0.75, 1,@-0.5,0),
   PAba (@-0.75, 0, @-0.25, 1, @0, 0),
   SinDesY (@-0.25, 0, @0, 1, @0.25, 0),
   PArri (@0, 0, @0.25, 1, @0.75, 0),
   Arr (@0.5, 0, @0.75, 1, @1,1)
];
invar ZDesU "" :-1.000000 (  ) 1.000000 [
   Atras (@-1, 1, @-0.75, 1,@-0.5,0),
   PAtras (@-0.75, 0, @-0.25, 1, @0, 0),
   SinDesZ (@-0.25, 0, @0, 1, @0.25, 0),
   PAdel (@0, 0, @0.25, 1, @0.75, 0),
   Adel (@0.5, 0, @0.75, 1, @1,1)
];
```

Conjuntos difusos de salida

```
invar XNDesU "" :-1.000000 (  ) 1.000000 [
   Izq (@-1, 1, @-0.75, 1,@-0.5,0),
   PIzqu (@-0.75, 0, @-0.25, 1, @0, 0),
   SinDesX (@-0.25, 0, @0, 1, @0.25, 0),
   PDere (@0, 0, @0.25, 1, @0.75, 0),
   Der (@0.5, 0, @0.75, 1, @1,1)
];
invar ZNDesU "" :-1.000000 (  ) 1.000000 [
   Atras (@-1, 1, @-0.75, 1,@-0.5,0),
   PAtras (@-0.75, 0, @-0.25, 1, @0, 0),
   SinDesZ (@-0.25, 0, @0, 1, @0.25, 0),
   PAdel (@0, 0, @0.25, 1, @0.75, 0),
   Adel (@0.5, 0, @0.75, 1, @1,1)
];
invar YNDesU "" :-1.000000 (  ) 1.000000 [
   Abajo (@-1, 1, @-0.75, 1,@-0.5,0),
   PAba (@-0.75, 0, @-0.25, 1, @0, 0),
   SinDesY (@-0.25, 0, @0, 1, @0.25, 0),
   PArri (@0, 0, @0.25, 1, @0.75, 0),
   Arr (@0.5, 0, @0.75, 1, @1,1)
];
```

Reglas difusas

```
$Base de reglas para el movimiento en X
if XPosU is DIzq and XDesU is Izq then XNDes is PIzqu;
if XPosU is DIzq and XDesU is PIzqu then XNDes is PIzqu;
if XPosU is DIzq and XDesU is SinDesX then XNDes is Izq;
.
.
.
```

Para el eje Z se tienen las siguientes reglas:

```
if ZPosU is MuLejos then ZNDes is Ade;
if ZPosU is Lejos then ZNDes is Ade;
if ZPosU is MLejos and ZDesU is Atras then ZNDes is PAde;
...
```