Ambientes virtuales particionados habitados por agentes autónomos y con interacción con el usuario basada en visión

Isaac Rudomín Goldberg PhD.
Fis. Saskia de Winter
Ing. Montserrat Morales

Septiembre 13 1997

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS ESTADO DE MÉXICO
Km 3.5 Carretera Lago de Guadalupe Atizapan, Estado de Mexico, Mexico
Teléfono: 326-5613 Fax: 326-5789
email: rudomin@campus.cem.itesm.mx
WWW: http://journey.cem.itesm.mx/isaac.html

Resumen: Este artículo describe una arquitectura para ambientes virtuales animados y distribuídos, accesados simultáneamente por varios usuarios conectados a través del Internet. Los usuarios podrán interactuar al mismo tiempo y de manera colaborativa. Este mundo virtual estará habitado por creaturas autónomas (agentes), construídas utilizando partículas y resortes, y sujetas a fuerzas simuladas. Estas creaturas contarán con visión sintética para la recolección de información y la toma de desiciones.

Por eficiencia, los ambientes virtuales estarán divididos en distintas regiones, cada una controlado por servidores distintos. Esto causa varios problemas que se han tenido que resolver: la consistencia del ambiente y el ajuste a los algoritmos de visión sintética para funcionar en un ambiente distribuído partido en regiones. Además se contempla la interacción con el usuario utilizando una cámara real. Una parte importante del proyecto es que se implantará utilizando Java y VRML.

Palabras Clave: Ambientes Virtuales, Agentes Autónomos, Modelado Basado en Física, Visión Sintética, Simulación Distribuída Interactiva, VRML, Java.

Ambientes virtuales particionados habitados por agentes autónomos y con interacción con el usuario basada en visión

1 Introducción

En 1993, Demetri Terzopoulos, Xiaoyuan Tu et al [1, 2, 3] de la Universidad de Toronto presentaron por primera vez Go Fish!, aplicación que simula el movimiento hidrodinámico de peces. Los peces se mueven en forma realista porque están basados en un sistema de partículas y resortes que reaccionan ante las leyes de la física. Ya para 1996, habían implantado algoritmos de visión activa para la foveación y detección de objetivos de interés, así como de objetos en movimiento, entre otras cosas. El proyecto también incluye un proceso de aprendizaje de habilidades motrices difíciles, por ejemplo, para que un delfín haga una pirueta o una cabriola en el aire.

En 1995, Pattie Maes, Trevor Darrel, Bruce Blumberg y Alex Pentland del MIT Media Lab presentaron un sistema llamado "ALIVE" [4]. Este sistema consiste en una interacción inalámbrica y de cuerpo completo con agentes autónomos. El sistema utiliza una cámara de video para recolectar la información de un usuario que se coloca frente a una pantalla. En esta pantalla, el usuario observa la escena hecha por computadora, el agente autónomo con el que interactúa y una incrustación de si mismo en tiempo real. Cada agente autónomo consta de un conjunto de sensores virtuales que, en este caso, es una grupo de rayos que se disparan en un plano 2D para obtener las intersecciones. Este método es precisamente la idea fundamental para la visión por computadora.

Cada uno de estos proyectos se caracteriza por haber sido innovador en alguna parte de su implantación. El grupo de investigadores de la Universidad de Toronto han llevado este proyecto durante casi 4 años, y comenzó siendo un proyecto muy sencillo para llegar hasta implantar peces con un sistema de foveación. Sin embargo, a diferencia de "ALIVE", el usuario solamente observa, y no interactúa con el medio. Por otra parte, "ALIVE" incluye una cámara real para detectar los movimientos de un usuario, permitiendole a este interactuar. Así, surge la idea de implantar un proyecto en donde se combinen las dos ideas principales de estas investigaciones: un mundo virtual con agentes autónomos que posean la **habilidad de ver**, y que se aproveche la **posibilidad de interactuar** con los agentes dependiendo de lo que "ven".

En todos estos proyectos, hay un único usuario. Una de las tendencias actuales es el de contar con sistemas de "Realidad Virtual Distribuída"; es decir, un mundo simulado que es observado al mismo tiempo por varios usuarios, en distintas estaciones de trabajo y de manera simultánea. Hay varias formas de hacer esto, y en este artículo discutiremos los protocolos y arquitecturas necesarios para construir un mundo virtual multiusuario que sea lo mas eficiente posible con los recursos disponibles. La solución propuesta es un sistema que consiste en clientes y de un multi-servidor. El mundo virtual se subdivide en regiones, y cada región es controlada por un servidor distinto, que puede comunicarse con los servidores vecinos, y con los clientes que en ese momento están dentro de la región.

Por otra parte, es importante notar que el adoptar esta arquitectura tiene consecuencias sobre muchos aspectos del mundo virtual. Hay que garantizar consistencia entre lo que ven los diversos usuarios, y también permitirles ver partes de regiones vecinas sin darles acceso a toda la información detallada de estas regiones. Además, los agentes tienen que considerar algoritmos de visión sintética que tomen en cuenta que no todo es directamente accesible en su región, pero puede ser importante

para ellos.

Además, nuestro sistema incluye una cámara real que permitirá la interacción de los diversos usuarios con los agentes. La cámara tomará ciertas posiciones de las manos del usuario, y estas posiciones son los diferentes comandos para comunicarse con los agentes autónomos en su mundo virtual. Hemos escogido también utilizar, al menos inicialmente, peces construidos a partir de partículas y resortes, sujetos a las leyes de la física.

Una parte que es importante enfatizar es que, por el momento, el comportamiento de los agentes es muy sencillo. Esto se refiere a traslaciones y rotaciones sencillas. Reaccionan de acuerdo a lo que "ven" y pueden cambiar de rumbo. Se tienen contemplados comportamientos mucho mas elaborados, pero no se han hecho aún, y por tanto quedan fuera de este artículo. También, es importante mencionar que se describe un proyecto que no está terminado, por lo que se reportan resultados parciales. Esto no es sorprendente dada la magnitud del proyecto.

Por último, es importante hacer notar que se quiere que este sistema funcione a través de internet. Los clientes son estaciones de trabajo o computadoras personales, sin importar que sistema operativo tienen. Es por ello que se ha elegido utilizar lenguajes independientes de plataforma, adecuados para internet, VRML [5] y Java [6]. Estos dos lenguajes, usados en conjunto permiten hacer simulaciones gráficas tridimensionales multiusuario.

Resumiendo, el sistema que describe este artículo consta de:

1. Implantación de un sistema de agentes autónomos de peces con comportamiento. El movimiento de los peces está controlado por partículas que permiten un movimiento natural y realista al pez.

2. Implantación de un sistema de visión para cada pez para el reconocimiento del mundo que le rodea. Esta información permitirá al agente tomar decisiones y realizar los movimientos acordes como alejarse o acercarse. El sistema de visión consiste en la generación de dos imagenes por pez. Cada imagen será analizada para obtener los puntos que concuerden y poder generar coordenadas en 3D de los objetos observados. El pez debe decidir cómo reaccionar posteriormente.

3. Los usuarios virtuales se pueden mover entre las regiones. Ellos observan cómo reaccionan los peces, pero no analizan las imagenes generadas.

4. Implantación de una cámara física para la interacción del usuario con el mundo acuático. Desarrollo del software necesario para reconocer los movimientos de las manos. Este sistema es el lenguaje de comunicación con los peces, por lo que también debe poder "ver" lo que el usuario está haciendo.

5. Todos los pasos anteriores deben funcionar en un sistema distribuído partido en regiones. Esto significa que el acuario se controla mediante el multi-servidor con regiones. Cada región debe avisar a la región vecina que un nuevo pez acaba de ingresar. Cada región sabe cuántos peces y cuántos usuarios contiene. Esto se explica en mas detalle en seguida.

2 Arquitectura del mundo virtual: Dividiendo en regiones

El término de "Realidad Virtual Distribuída" se refiere a un mundo simulado que no corre únicamente en una computadora, sino en muchas. Las computadoras estan conectadas a través de redes,

y los usuarios son capaces de interactuar en tiempo real, compartiendo el mismo mundo virtual.

Para diseñar la arquitectura de distribución, se hicieron comparaciones entre los protocolos estándares y arquitecturas existentes en el campo de realidad virtual distribuída.

- *El particionamiento de un mundo virtual*

Los mundos o ambientes virtuales se convierten cada día en plataformas para escenas cada vez mas complejas. Los mundos virtuales incluyen "seres" que tienen comportamiento propio, y que entre otras características utilizan diferentes texturas, materiales, luces y movimientos de cámara para aumentar su parecido con la realidad. Cada uno de estos elementos representa mayor número de cálculos que obviamente vuelven más lento al sistema. Para diseñar la distribución del mundo virtual, tomando en cuenta los múltiples modelos estudiados, se analizaron diferentes posibilidades que pueden resumirse como sigue:

1. `Modelo Centralizado`
   La estación de trabajo que inicia la animación tiene el control del mundo simulado, y distribuye esta información a los demas usuarios. Puede utilizarse comunicación tipo *unicast*, lo cual satura la red y convierte al servidor en un cuello de botella. Para evitar la saturación, es posible utilizar también comunicación tipo *broadcast*, pero esto no hace nada por reducir la carga del servidor. Sin embargo, la ventaja de este modelo es que hay consistencia entre lo que ven todos los usuarios, pues consultan al mismo servidor.

2. `Modelo Totalmente Descentralizado o Distribuido`

   Cada máquina tiene una copia del mundo virtual. Al efectuar cualquier modificación a este mundo, se manda una notificación a los demás para que se actualicen.

   La ventaja es que no hay un servidor central que pueda convertirse en un cuello de botella.La desventaja es que para mantener la consistencia es necesario mandar una gran cantidad de mensajes. Esta opción no es buena cuando se tienen muchos usuarios, o mundos que cambien frecuentemente, ya que el intercambio de mensajes puede llegar a saturar la red. Para evitar esto, puede utilizarse una comunicación tipo *multicast*, pero esta no está aún diponible de manera general en internet, por lo que, al menos por el momento, no puede utilizarse esta arquitectura.

3. `Modelo Parcialmente Descentralizado`

   El mundo se divide en regiones, y hay varias computadoras funcionando como servidores de regiones. El punto delicado a tratar es cómo se va a dividir este mundo, es decir, en cuántas regiones, qué van a contener, qué tanto van a manejar, y qué tanto cálculo se va a hacer en estas regiones. La comunicación entre cada región depende de la información que cada una maneja. La principal ventaja es que no hay un servidor central, aunque si varios servidores. Los mensajes solo se mandan a quienes tienen interés en ellos. Es una combinación entre las dos opciones anteriores. La principal desventaja es que requiere de un protocolo complejo.

Después de un análisis como el anterior, se decidió implantar la opción de dividir el mundo por regiones y utilizar el modelo parcialmente descentralizado; es decir, sin servidor central.

Un mundo virtual dividido en regiones aprovecha el poder de cálculo de las unidades centrales de procesamiento de cada estación de trabajo, evitando así la lentitud. Cada región cuenta con una lista de operaciones que debe realizar para controlar su pequeño mundo, y los datos se transmiten entre las regiones vecinas para su interacción entre sí.

La propuesta presentada en este artículo incluye:

Subdivisión del mundo virtual
El mundo está subdividido en regiones. En cada región, existen agentes (peces) y usuarios. A medida que aumenta el número de peces y de usuarios, es necesario subdividir mas el mundo para una configuración óptima. Existe un multi-servidor que calcula la posición de cada uno de los peces y de los usuarios por región. Además, cada una de las estaciones tiene una copia de la región en donde el usuario actual se encuentra. De este modo, el multi-servidor y las estaciones de trabajo calculan con diferentes algoritmos las nuevas posiciones según el movimiento natural de cada pez y según el movimiento controlado del usuario. Si las posiciones del multi-servidor y de la estación de trabajo para un mismo pez difieren, entonces, la del multi-servidor es la escogida. Este proceso sirve para evitar una cantidad excesiva de mensajes entre el multi-servidor y las estaciones: solo se transmiten los datos cuando difieren por un umbral significativo.

Esta es una solución similar al *dead reckoning*. Este método consiste en tener copias de cada agente o usuario relevante en cada estación de trabajo, asi como su posición y velocidad. El dueño de cada agente calcula la posición real de este, pero además hace una predicción utilizando la posición y la velocidad. Por otra parte, las copias solo hacen la predicción, y la utilizan siempre y cuando el dueño no les mande un mensaje con una nueva posición y velocidad. De hecho, es cuando determina que ambas trayectorias, la real y la predicha son ya muy distintas. Este método es parte fundamental del protocolo estándar llamado DIS [7, 8, 9], muy utilizado en simulación interactiva distribuida para casos militares.

En nuestro caso, la información que se requiere en un mundo con física simulada incluye además la aceleración y las fuerzas que afectan al agente.

3 Visión Sintética en Ambientes Particionados

La visión por computadora se presenta típicamente como parte de un proyecto de robótica. De cierto modo, este proyecto es similar. Los agentes autónomos obtienen la información por medio de diferentes métodos de aprendizaje, y la percepción mediante visión sintética es un de ellos. El color, la forma y el movimiento se obtienen para definir si lo que se observa es comida, otro pez o un tiburón. Así, con la información recolectada, el pez puede decidir si se acerca o se aleja.

Así, se propone implantar un sistema de visión sintética para que un grupo de agentes autónomos puedan recolectar información del mundo virtual en el que viven.

Existen varias diferencias de la visión sintética de un solo mundo con respecto a un mundo particionado. Estas diferencias consisten básicamente en que no tenemos acceso directo a imágenes u objetos de otras regiones. Debemos entonces, pedir información de las regiones vecinas y generar imágenes compuestas de lo que un agente (o usuario) puede ver antes de comenzar a utilizar los algoritmos de visión usuales.

Además, pueden existir varios usuarios en remoto interactuando con el sistema, ya sea navegando a traves de las diferentes regiones que conforman el mundo, o bien mediante una cámara física por medio de la cual con ademanes se pueden comunicar con los peces.

Existen sistemas conocidos que han implantado visón sintética, sistemas distribuídos, interacción y colaboración, pero ningún sistema que conjunta todas estas características en un mismo proyecto, sobre todo el concepto de visión sintética en ambientes distribuídos partidos en regiones.

Recolección de datos:
La recolección de datos para este proyecto consiste en la generación de imágenes en 2D de aquello que el pez observa. Existen diferentes maneras de llevarlo a cabo. En Open Inventor [14], en el que se hicieron las primeras pruebas, se puede pedir que el sistema genere un dibujo fuera de linea (*offscreen render*) o también generar otros puntos de vista en ventanas nuevas; por ejemplo, tener dos ventanas pequeñas por separado que muestran lo que cada ojo del pez observa. Para los primeros prototipos, se llevaron a cabo otras pruebas: se disparan rayos desde cada ojo en un rango del 300 grados. La razón de este número tan específico es que realmente los peces en la naturaleza tienen ese ángulo máximo de visión. Cuando el rayo intersecta un objeto; es decir, cuando el rayo encuentra un pixel de color distinto a aquel del color del agua o del ambiente, entonces las dos imágenes se generan: exactamente lo que el pez ve, el ojo izquierdo y el derecho. Cada una de estas imágenes pasa a traves de un proceso de reconocimiento para determinar los objetos que se observan. Este procedimiento de generación de imágenes se conoce como esteresocopía, que es similar al proceso natural de la visión humana.

Dentro de nuestro mundo segmentado, cada pez vive en una región (digamos que es un cubo). Cada vez que el pez require de ver fuera de su región, debe pedir la información a la región vecina. Esto es equivalente a tener imagenes generadas con anterioridad que se proyectan en cada una de las caras de los cubos y que ademas se utilizan como fondo para generar y componer encima objetos de la nueva región: es como un papel calca, en el que se va acumulando lo que se va dibujando a cada nivel. En la mayoría de los casos, solamente será necesario generar y analizar dos de las seis caras del cubo, a menos de que la posición de pez requiera que se generen mas de dos caras; es decir, tres, y que obviamente depende del punto de vista del pez. El acuario está dividido en regiones. La imagen que se dibuja en la cara del cubo es la acumulación de las imagenes de las caras de los cubos vecinos. Dependiendo del rango de vista del pez se acumularán mas o menos imagenes en las caras de los cubos. Ver figura 1.

Análisis de las imágenes:
Una vez que el pez haya generado las imagenes de ambos ojos, cada una de éstas deben ser analizada para obtener los puntos y coordenadas significativas. Como se explicó con anterioridad, existen muchos procedimientos de segmentación y es necesario observar cuál es el más eficiente y el mas veloz. Se generan imagenes de 64 x 64 pixeles aunque el tamaño puede variar. El tamaño que se escogió se debe a las consideraciones del tiempo de render.

El usuario también es parte del acuario. Cada usuario puede vivir en una sola región a la vez, y se puede mover de región en región para observar las reacciones de los peces. Tanto los peces como los ususarios se pueden mover entre las regiones. De hecho, cada región contiene una lista de peces y de usuarios. La diferencia principal entre un pez y un usuario en cuanto a los cálculos internos es que el usuario ya puede ver y por lo tanto no es necesario hacer el análisis o la detección de contornos de las imagenes.

Hay muchos métodos que permiten de alguna manera obtener descripciones de objetos a partir de imágenes. Cuando estas imágenes son sintetizadas por una computadora en vez de provenir de una cámara, a pesar de utilizarse algoritmos similares, el proceso es conocido como visión sintética. De hecho, estas imágenes son mas controlables, por lo que se pueden utilizar algoritmos mas simples y obtener información suficiente para que un agente pueda tomar una desición. Como ejemplos de

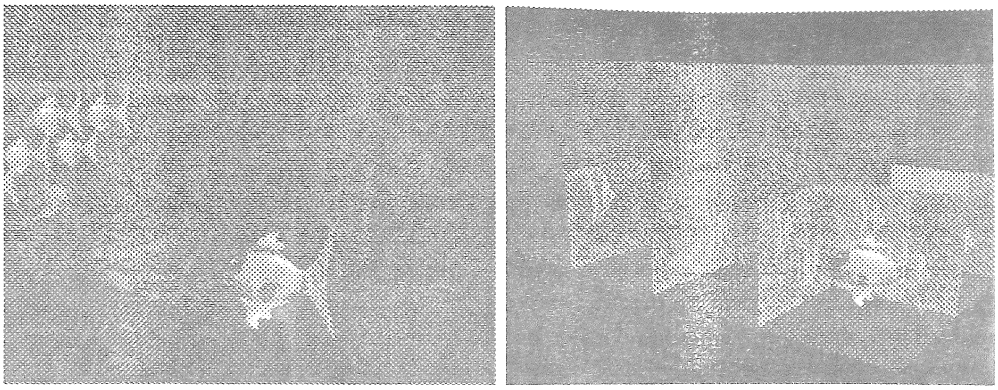

Figura 1: Cómo vive el pez en las regiones, Proyección en las paredes

estos procesos, que se han implantado, pero aún no son utilizados por la versión actual del sistema están la Detección de contornos y la Estereoscopía [10, 11, 12, 13].

4 Resultados y Trabajo a Futuro

Este artículo describe trabajo que está aún en proceso. A la fecha se ha hecho lo siguiente:

1. Se han estudiado las diversas alternativas de arquitectura. Después de diversas pruebas se decidió utilizar la arquitectura basada en servidores de regiones. Se evaluaron también varios lenguajes, y se ha decidido utilizar VRML 2.0 y Java debido a sus múltiples ventajas que incluyen multi-threading y la independencia de plataforma.

2. Se implantó una versión no distribuída del mundo de peces con física de partículas (en Open Inventor).

3. En otro trabajo se está avanzando en la percepción de los agentes autónomos. Por de pronto, los peces reaccionan en base a un mecanismo de colorimetría. Pueden ya reconocer ciertos colores y reaccionar distinto. Con el rojo se alejan, con el azul se acercan.

4. Se ha implantado una versión multiusuario, con servidor central, en Java y VRML 2.0 de:

   (a) DIS, con *dead reckoning, heartbeat*

   (b) *dead reckoning* modificado para mundos con física de partículas. Esta version se utilizará como plataforma para re-implantarla utilizando ya la arquitectura basada en regiones.

(c) En cuanto a la visión sintética, Open Inventor se usó para crear las primeras pruebas para lanzar los rayos y la generación de imágenes. Los peces tienen movimientos sencillos como translación y rotación. La cantidad de peces oscila entre cinco y siete y a medida que aumenta el número de peces, el sistema se vuelve mas lento. Esto sucede porque aún no es un sistema distribuído, y todo el cálculo se realizo en el mismo CPU (Silicon Graphics Indigo Extreme).

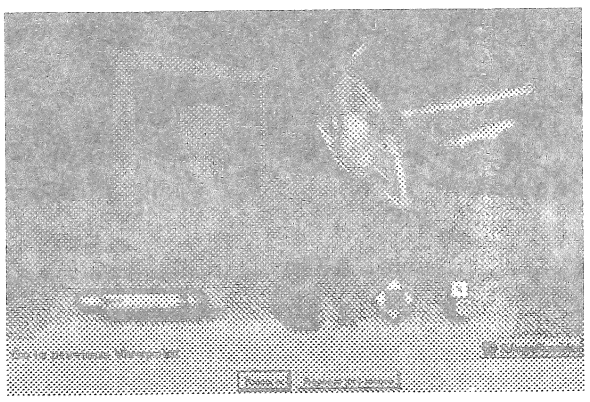

Figura 2: Un cilindro azul

Esta parte del proyecto es sobre visión e incluye dos partes importantes:

i. Los peces requieren de un sistema de percepción para observar el acuario en el que viven. A partir de la información que se recolecta, el pez "entiende" todo lo que observa y debe reaccionar como una agente autónomo y toma una decisión. Por ejemplo, el pez debe distinguir entre un depredador y comida. Si el sistema es efectivo y los peces son lo suficientemente rápidos, deben poder tomar la desición de escapar de un tiburón a tiempo. También, existen usuarios que navegan de región en región para observar el acuario.

ii. El sistema consta también de una cámara real conectada como perférico a la estación de trabajo. La cámara captura la imagen del usuario interactuando con el acuario. Por ejemplo, si el usuario levanta la mano izquierda, puede significar que el pez se acerque mientras que si es la mano derecha puede significar que el pez se aleje, tal y como sucede en un acuario real.

El sistema de percepción tiene su origen en los ojos de cada pez. Cada pez observa su mundo desde su propio punto de vista, y cada pez recolecta información distinta del mundo. El resultado de lo que cada pez observa son las imágenes en 2D de cada ojo. Es como si los peces tuvieran dos cámaras instantáneas en cada ojo. Los fotografías proveen la información para decidir cómo comportarse.

El sistema está parcialmente implantado en Open Inventor. Consiste en un pequeño banco de peces con movimientos simples de translación y rotación. Cuando el sistema determina que existe un objeto, se generan las imágenes de cada ojo. En las siguientes imágenes, se puede observar que existe una pequeña diferencia: el ojo izquierdo está ligeramente hacia la izquierda con respecto al ojo derecho. Este fenómeno sucede debido a la distancia entre los centros ópticos (disparidad). Una vez que las imágenes son analizadas, los agentes autónomos determinan si siguen nadando hacia adelante, o deben

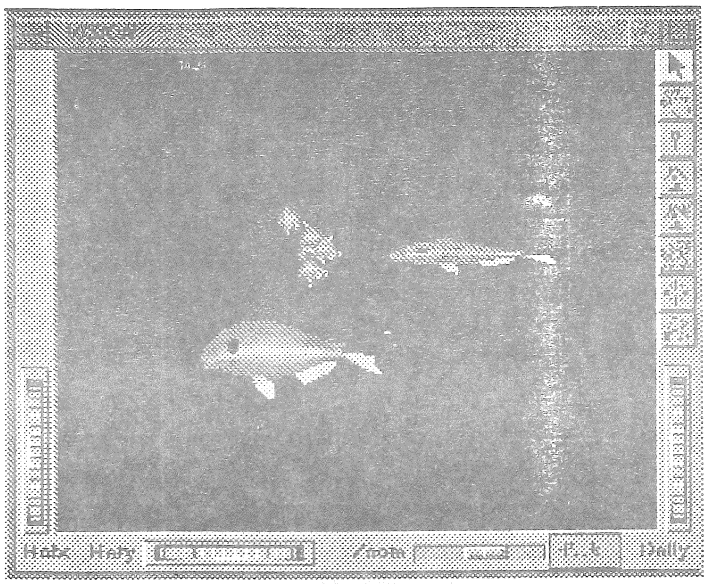

Figura 3: Implantación en Open Inventor

cambiar su rumbo. Estos algoritmos de detección de contornos y segmentación se deben transcribir aún a Java y VRML.

Un prototipo de cliente que un usuario utilizaría para interactuar con el acuario se ha implantado en Java y VRML. Esta interfaz presenta por el momento un pez observando una pantalla (como si fuera una de las caras de otra región). Esta pantalla muestra figuras que van cambiando al azar: un cilindro azul, una esfera amarilla, un cono rojo y una imagen con peces. El pez observa la pantalla y reacciona según lo que ve. El pez tiene un mecanismo que actualmente funciona en base a la colorimetría de lo que observa. Esto significa que al pez le da igual, por el momento, reaccionar ante un cilindro rojo, un cubo rojo o un tiburón rojo. El siguiente paso es generar las formas para determinar los comportamientos adecuados. Estos algoritmos aún no se implantan.

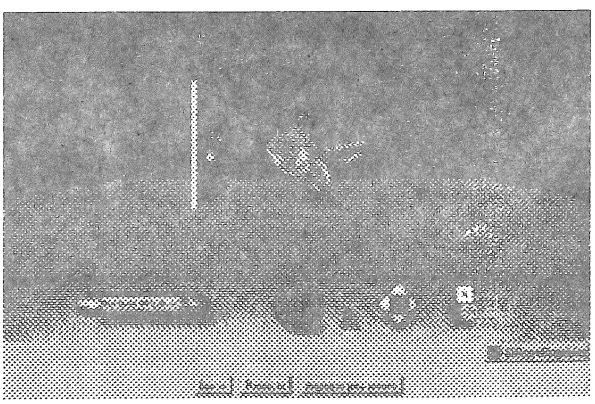

Figura 4: Pez viendo la televisión y reaccionando a las imágenes

La implantación de la cámara física también está en proceso. La nueva estación de trabajo de Silicon Graphics, la O^2, tiene un sistema especial de memoria para manejar la

entrada de video en tiempo real. Los filtros de Sobel y de Prewitt ya han sido implantados para la detección del movimiento de las manos. Tambien se han implantado otros filtros como el filtro de las derivadas parciales al cuadrado. Este tipo de filtro, junto con el filtro de detección de contornos de Laplace necesitan que exista movimiento en la escena para que puedan determinar los contornos. Los filtros de Sobel y de Prewitt funcionan aún cuando el usuario mantenga fija la mano. Si el usuario mantiene la mano en la misma posición, el filtro de Sobel es el mejor, pero si el usuario mueve contínuamente la mano, entonces el filtro de Laplace es mas preciso. Parte de este trabajo puede consultarse en trabajo nuestro previamente presentado en el congreso Computación Visual 1997[15, 16].

4.1 ¿ Qué falta por hacer?

Como ya mencionamos, se está trabajando en implantar el mundo de peces en VRML 2.0 y Java, lo que va muy avanzado. Hay ya una versión que incluye un pez, implantado con partículas y resortes, que puede analizar una imágen y moverse según que ve en la imágen.

Además, mediante diferentes puntos de vista de cámaras dentro de VRML se puede observar exactamente lo que ve el ojo izquierdo o el ojo derecho del pez, esto parte ya también está implementada. Es curioso poer "ver" lo mismo que el pez está "viendo". Lo que le falta hacer en esta parte es la implantación final de la arquitectura propuesta, asi como utilizar algoritmos de visión mas sofisticados, en particular la propuesta de algoritmo de visión en el mundo subdividido en regiones. Se está experimentando, y de hecho ya se puede hacer esto último en OpenInventor.

Existen dos problemas críticos en VRML:

i. Los browsers de VRML no cuentan con un *ofscreen renderer*. Esto significa que a partir de VRML no se pueden generar imagenes de lo que el pez ve para ser analizadas, tal y como se ya implementó en Open Inventor. Habrá que buscar otro tipo de solución.

ii. No hemos podido aún resolver el problema de pegar texturas móviles creadas en el momento a las caras del cubo que representa la región (es decir: a hacerlo con suficiente velocidad y sin comprar productos que lo permiten). Por de pronto, la velocidad con la que se ha logrado este proceso aún sigue siendo demasiado lento y no con la cantidad de información necesario como para poder "creer" que es la información de la región vecina.

Estamos trabajando en este problema, aunque parte del problema debe solucionarlo mas bién el desarrollador del CosmoPlayer.

También falta implementar un sistema eficiente de detección de ademanes para la cámara física. Esta parte se está desarrollando en Open Inventor, y en Open GL porque son los lenguajes con los que se puede accesar el buffer de memoria de la cámara de la O^2. A pesar de que ya se tienen filtros muy eficientes, aún es necesario controlar por regiones la imagen que se analiza para determinar si la mano que se levantó fue la izquierda o la derecha. Probablemente un punto delicado en todo el proyecto será la unión de todos los módulos, considerando que se están usando por lo menos 4 diferentes lenguajes de programación. Este trabajo se puede extender en cualquiera de las areas que se manejan en el mismo, como lo puede ser el modificar los agentes, el protocolo de comunicación, o

bien el sistema de regiones, el metodo de resolución de ecuaciones de segundo grado.

Se pueden implantar diversas caracteristicas, y complejidad a la arquitectura de division de regiones, cambiar el protocolo de comunicacion, hacer que los agentes sean inteligentes, etc.

5 Bibliografía

Referencias

[1] Terzopoulos Demetri, Tu Xiaoyuan. *Perceptual Modeling for the Behavioral Animation of Fishes.* Department of Computer Science, University of Toronto.

[2] Terzopoulos Demetri, Tu Xiaoyuan, Grzeszczuk Radek. *Artificial Fishes: Autonomous Locomotion, Percepition, Behavior, and Laerning in a simulated Physical World.* Department of Computer Science, University of Toronto.

[3] Terzopoulos Demetri, Rabie Tamer, Grzeszczuk Radek. *Perception and Learning in Artificial Animals.* Department of Computer Science, University of Toronto.

[4] P. Maes, T. Darrell, B. Blumberg, and S. Pentland, *The ALIVE System: Full-Body Interaction with Autonomous Agents* Proceedings of the Computer Animation '95 Conference, Geneva, Switzerland, pp. 11-18, IEEE Press, April 1995.

[5] Josie Wernecke, *The VRML 2.0 Handbook* Addison Wesley 1997

[6] Flanagan David. *Java in a Nutshell.* OReilly Associates, Inc. 1996

[7] *NPSNET IV.7J System Overview.* June, 1995

[8] R. Macedonia Michael, P. Brutzman Donald, J. Zyda Michael, R. Pratt David, T. Barham Paul, Flaby John, Locke John. *NPSNET: A MULTI-PLAYER 3D VIRTUAL ENVIRONMENT OVER THE INTERNET.* Naval Postgraduate School, Departament of Computer Science. Symposium on Interactive 3D Graphics. April, 1995.

[9] Rich Gossweiler, Robert J, Laferriere, Michael L. Keller, Randy Pausch.
An Introductoy Tutorial for Developing Multi-User Virtual Environments. University of Virginia, Computer Science Department.

[10] Ballard Dana H., Brown Christopher. *Computer Vision.* Prentice-Hall. New Jersey. 1982.

[11] Computer Vision for Computer Graphics. Course Notes for SIGGRAPH 95.

[12] Faugeras Olivier. *Three-Dimensional Computer Vision- A Geometric Viewpoint.* Mit Press. Cambridge, Massachusetts. 1993

[13] Fernald Rusell D. *The physiology of Fishes. Chapter 6 Vision.* CRC Press. Boca Raton, FL. 1993.

[14] Wernecke Josie. *The Inventor Mentor. Programming Object-oriented 3D Graphics with Open Inventor, Release 2.* Addison-Wesley Publishing Company. 1994

[15] Montserrat Morales, Isaac Rudomín *Arquitectura para realidad virtual distribuída con física simulada, poblada de agentes autónomos* COMPUTACIÓN VISUAL 97

[16] Saskia de Winter, Isaac Rudomín *Synthetic Computer Vision for Autonomous Agents in Distributed Partitioned Environments* COMPUTACIÓN VISUAL 97