

Entornos virtuales 3D clásicos e inteligentes: hacia un nuevo marco de simulación para aplicaciones gráficas 3D interactivas

Miguel Lozano, Carlos Calderón

Departamento de Informática, Universidad de Valencia,
Dr.Moliner 50, (Burjassot) Valencia (Spain)
{Miguel.Lozano}@uv.es

Architectural Informatics Group, University of Newcastle,
Claremont Road, Newcastle, UK
{carlos.calderon}@newcastle.ac.uk

Resumen

En este artículo analizamos la evolución experimentada por los Entornos Virtuales 3D (EV3D) con el énfasis puesto en las cuestiones relativas a su comportamiento como aplicaciones informáticas en tiempo de ejecución. En primer lugar, analizamos los modelos computacionales que soportan la mayoría de entornos virtuales 3D, poniendo de manifiesto las carencias comportamentales de los entornos dependientes del modelo *clásico* de simulación gráfica. En base a estas carencias, analizamos las directrices de diseño de los nuevos Entornos Virtuales Inteligentes (EVI3D), orientados a proporcionar distintas capas de IA (Inteligencia Artificial) que pueden ser embebidas en el EV3D de cara a mejorar sus capacidades interactivas. Finalmente, concluimos argumentando que los entornos y consecuentemente las tecnologías que los soportan, basados en la distribución de eventos de forma asíncrona, proporcionan actualmente un marco adecuado para la nueva generación de EVI3D.

Palabras clave: Entornos Virtuales (Inteligentes) 3D (EV3D, EVI3D), Sistemas Inteligentes, Agentes Inteligentes 3D (3DIVA), Sistemas Interactivos 3D.

1. Introducción

El concepto de entorno virtual en 3D ha evolucionado significativamente en los últimos años. Esta bien documentado [15], que los mundos virtuales tienen su origen en la simulación militar y en concreto en los simuladores de vuelo, donde el principal problema consiste en extraer de la base de datos visual (presumiblemente grande) el mundo

visible en cada instante en función de la posición del observador o cámara virtual, en el escenario simulado. Consecuentemente, la comercialización de esta tecnología para uso civil dio origen al concepto de Realidad Virtual que todavía hoy perdura: gráficos 3D en entornos inmersivos que usan I/O artefactos como guantes, cascos, etc. en busca de mayores grados de interacción con el ambiente virtual.

En esta última década, a la vez que la Realidad Virtual se comercializaba en entornos no necesariamente inmersivos, desde el sector de los juegos por computador apareció un nuevo tipo de EV3D interactivo, conocido como motor de juego (*game engine*). Estos sistemas, pronto se mostraron capaces de mantener un grado de calidad visual equivalente a las aplicaciones de simulación (civil/militar), obteniendo además mayores grados de interactividad gracias a los fuertes requerimientos que poseen como plataforma de soporte a juegos de distinta naturaleza.

Así, hemos llegado a la situación actual, donde la excelente calidad gráfica alcanzada por la tecnología gráfica 3D, por un lado refuerza la metáfora de mundo virtual compartido (de naturaleza interactiva), y por otro, potencia las expectativas comportamentales de los elementos (objetos y agentes) que típicamente lo habitan.

Por consiguiente, hoy en día, científicos e investigadores en el área de entornos virtuales 3D interactivos centran su atención en la creación de los denominados: Entornos Virtuales Inteligentes (EVI) [17]. Esto es, entornos virtuales que surgen de la intersección **IA** \cap **gráficos 3D en tiempo real** y en los que sus capacidades comportamentales e interactivas son ensalzadas a través de la inclusión de técnicas de inteligencia artificial (ver sección 3). El área más investigada de esta intersección es el campo de agentes virtuales inteligentes 3D (3DIVA) y, por consiguiente, muchos de los ejemplos del artículo hacen referencia a este área.

En este artículo revisamos los modelos computacionales que dan soporte a la mayoría de entornos gráficos 3D, poniendo el énfasis en la simulación de sus aspectos comportamentales en tiempo de ejecución. En la segunda sección analizamos los modelos computacionales (geométrico, comportamental y de interacción) que conforman un EV3D. Fruto de esta revisión exponemos las principales deficiencias (comportamentales y de interacción) de los modelos *clásicos* de simulación gráfica. Finalmente, en la segunda parte del artículo, describimos las principales mejoras comportamentales e interactivas que ofrecen los nuevos EVI3D, a partir de los distintos ejemplos de aplicación expuestos.

2. Modelos computacionales

Independientemente del ambiente interesado en su modelización virtual, desde un punto de vista exclusivamente computacional, los EV3D están compuestos por tres modelos que conformarán la aplicación informática en tiempo real: un modelo geométrico, un modelo comportamental (en tiempo de ejecución) y un modelo de interacción con el usuario [14].

2.1. Modelo geométrico

Es el encargado de atender al bajo nivel gráfico, donde por un lado, los distintos tipos de formatos gráficos empleados, junto con el modelo interno utilizado para el lanzamiento de las órdenes de dibujo o *render* (grafo de escena, ordenación de primitivas gráficas o *display lists*, etc), resumen las propiedades elementales para la visualización de cualquier EV3D.

Tradicionalmente, estos entornos son descritos en base a la colección de primitivas poligonales, líneas, texto, superficies, etc., que constituyen la información espacial visualizada en 3D. Gracias a la implantación de hardware dedicado al procesamiento de dicha información visual, los sistemas gráficos de tiempo real, hoy en día, son capaces de dibujar millones de polígonos por segundo, lo que resulta ser una tasa de rendimiento bastante razonable, atendiendo a las numerosas aplicaciones gráficas aparecidas en los últimos años.

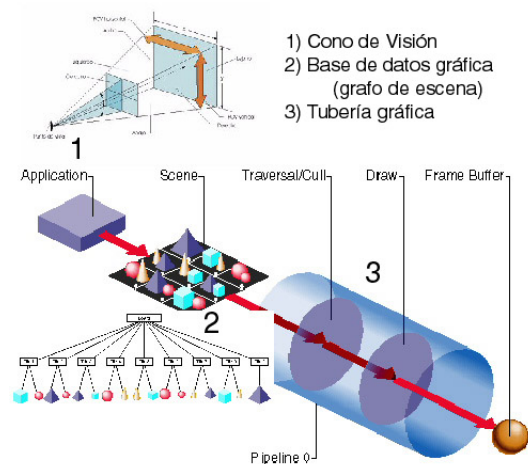


Figura 1. Modelo geométrico clásico = Grafo de escena + Tubería gráfica (app+cull+draw)

El modelo geométrico más básico consiste en lanzar directamente las órdenes de dibujado de cada uno de los polígonos que conforman la escena 3D. Las así listas generadas son procesadas por la tubería gráfica, dando como resultado la imagen 2D captada por una cámara virtual, localizada en algún punto del escenario 3D.

A medida que los EV3D fueron ganando en complejidad, este modo de dibujado resultaba insuficiente para la organización y mantenimiento de las escenas simuladas, por lo que apareció una nueva estructura, conocida como **grafo de escena** o *scene graph* (figura 1). A diferencia del modo de dibujado básico, los entornos basados en grafos de escena obtienen cada imagen sintética tras realizar un recorrido (*traversal*) sobre dicho grafo, normalmente acíclico y dirigido. Durante este recorrido, se generan y ordenan las listas de objetos 3D que caen dentro del campo de visión de alguna cámara virtual, para finalmente ser introducidas en la tubería gráfica. La organización espacial implícita en el grafo de escena favorece la organización de la base de datos visual, lo que resultará muy beneficioso en el proceso de recorte o *culling* asociado a cada orden de dibujado, y finalmente supondrá un aumento en la tasa de imágenes por segundo generadas.

Actualmente, muchos de los escenarios de simulación 3D dedicados al entrenamiento de tareas y realidad virtual aplicada, utilizan el modelo geométrico basado en la tubería gráfica que muestra la figura 1. El lenguaje de modelado VRML [1], y otras APIs (*Application Programmer's Interface*) como OpenGL-Performer [12], OpenInventor [10] o Java3D [11], son ejemplos de EV3D basados en grafos de escena (para una revisión más detallada consultar [14, 3]).

Desde un punto de vista exclusivamente geométrico, la integración de cuerpos articulados en el modelo anterior (por ejemplo, actores 3D de aspecto humanoide), no plantea en principio, serias dificultades. Esta integración pasa por la definición del esqueleto de soporte al actor 3D, donde distintos nodos (matrices) de transformación (DCS, *Dinamyc Coordinate Systems*) son frecuentemente utilizados para el diseño y control de personajes articulados 3D [19, 22, 20]. Una vez diseñado y modelado el personaje, éste es integrado en el EV3D *colgando* su estructura jerárquica del grafo de escena y pasando así a formar parte del escenario 3D.

En ocasiones, los humanoides 3D permiten la deformación de ciertas partes de su esqueleto

geométrico [2]. Típicamente estas deformaciones son tratadas mediante nodos que, incluidos en el grafo de escena, son asociados a ciertas articulaciones del esqueleto del actor. Por ejemplo, la geometría del cuello de un humanoide 3D (vértices que unen su cabeza con el resto del esqueleto) puede depender de este tipo de nodos de deformación (técnica conocida como *skinning*), lo que aportará finalmente mayor realismo visual a los movimientos de cabeza, derivados por ejemplo de la atención visual. Estas técnicas de deformación tienen costes computacionales nada despreciables, por lo que su control o comportamiento en esqueletos complejos (como son los humanoides 3D) debe ser controlado. Idealmente, estos nodos deberían activarse únicamente al producirse cambios de posición en las partes móviles que articulan y de las que dependen directamente. Es decir, la deformación del cuello sólo es necesaria cuando la cabeza esté girando, por ejemplo, en busca de algún objeto. De esta forma, para facilitar el autocontrol del modelo geométrico de un actor 3D, se requerirá algún modelo de comunicación entre los nodos del grafo al cual pertenece. Así, en tiempo de ejecución un nodo podrá conocer el estado de las partes del esqueleto que necesite para autoregularse.

Este tipo de cuestiones relativas al manejo de información en tiempo de ejecución, consecuencia directa de algún cambio producido en el EV3D (grafo de escena), es uno de los puntos principales de este artículo y nos conduce directamente hacia el modelo comportamental de los EV3D, revisado en la siguiente sección.

2.2. Modelo comportamental

Este modelo trata el comportamiento del EV3D como aplicación informática en tiempo de ejecución, es decir, atiende al comportamiento dinámico (cambios) de todos los objetos 3D que el EV3D contiene.

Conviene aclarar que el comportamiento aquí considerado, típicamente se refiere al mantenimiento en tiempo de ejecución de propiedades elementales, como posición, orientación, color, etc. de todo elemento (objeto u actor) situado en el EV3D. No debe por tanto confundirse con el comportamiento autónomo típicamente asociado a la toma de decisiones de distintos 3DIVA, u objetos autónomos que pueden formar parte de la simulación, cuestión que cae fuera del presente análisis.

En este artículo denominamos **EV3D clásicos** a aquellos cuyo modelo comportamental es altamente dependiente del modelo geométrico (grafo de escena), pasando así a depender del mismo. Un ciclo de simulación de un EV3D clásico suele dividirse en tres fases, que serán organizadas de igual modo que el hardware gráfico, es decir, a modo de tubería. La primera fase o de aplicación (app) está dedicada a la simulación, es decir, a producir todos los cambios necesarios en el entorno (posiciones de objetos móviles, cambios en la iluminación, etc). Una vez finalizada comenzará el recorrido por el grafo, el cual se encargará tanto del test de visibilidad (cull) como de compilar las listas de dibujo (*display list*).

De esta forma, y una vez cargada la escena completa, la aplicación recibirá el grafo de escena resultante, el cual podrá manipular desde ese momento (mediante el API asociado), para poder simular los cambios necesarios. Los principales nodos implicados en el modelo comportamental de los EV3D clásicos varían desde los nodos de LOD (*Level Of Detail*) capaces de seleccionar, cambiar y/o fusionar la apariencia geométrica de un objeto 3D en función de su distancia a la cámara, pasando por los nodos conmutadores (*SWITCH*), hasta los más recientes *Billboards*, capaces de orientar su geometría en función del ángulo abierto de nuevo con la cámara ¹. Los DCS ya comentados o nodos de transformación, son normalmente los encargados de trasladar, rotar o escalar su subárbol (nodos geométricos) asociado (ej: animación de los pájaros de la figura 2).

Sin embargo, tal y como muestra esta figura, los cambios (en este caso de posición) únicamente son llevados a cabo dentro de la zona visible, quedando el resto de cambios *podados* gracias al proceso de recorte (cull), llevado a cabo por el grafo de escena.

La reducción del ámbito de la simulación al mundo visible en cada ciclo (app-cull-draw), conllevará claros problemas comportamentales, ya que se impide cualquier tipo de cambio o interacción más allá del área visible.

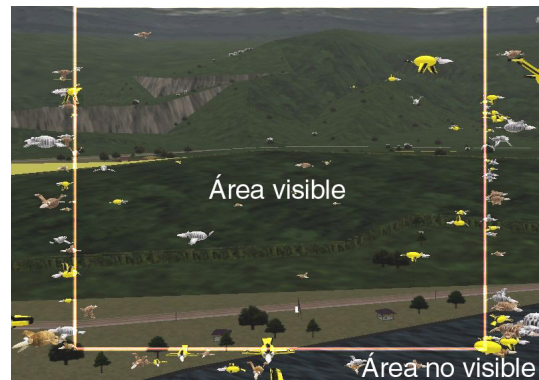


Figura 2. Visualización de comportamientos dependientes del grafo de escena.

Además de este importante problema, que podría ser parcialmente tratado mediante la relajación del proceso de recorte (o añadiendo niveles de detalle comportamentales), los EV3D clásicos no poseen ningún mecanismo general para establecer rutas de información entre los nodos del grafo de escena, lo que finalmente les obligará, desde el punto de vista comportamental, a ser considerados como **entornos basados en el tiempo** (*time based environments*) donde típicamente existirá un tiempo absoluto, inicializado al comienzo de la simulación y utilizado para fijar el inicio y el fin de los cambios producidos. En este sentido, por ejemplo VRML posee nodos interpoladores, capaces de animar distintos grados de libertad en función del tiempo y de los valores clave (keyframing) definidos (estrategia utilizada frecuentemente en los sistemas motores de distintos actores 3D).

Por tanto, los comportamientos soportados en este tipo de escenarios, pueden ser considerados como **independientes del estado de su entorno**, ya que ningún nodo del grafo tendrá la posibilidad de autogobernarse en función del estado de otro nodo vecino, pues no poseen (en principio) rutas de conexión entre ellos. De este modo, si se desea contemplar interacciones básicas entre objetos y 3DIVA (ej.: coger o dejar objetos, abrir o cerrar una puerta, ventana, etc...) en este tipo de entornos, éstas deberán ser programadas de forma local ², en perjuicio de importantes factores como la reusabilidad y escalabilidad del EV3D.

Como consecuencia, en tiempo de ejecución se obtienen EV3D completamente centrados en el campo de visión del usuario y estáticos desde un punto de vista comportamental.

¹Suelen emplearse en la representación de objetos complejos con cierto grado de simetría vertical, como árboles etc.

²Estableciendo hilos de comunicación *ad hoc* entre los nodos del grafo de escena implicados.

En busca del citado dinamismo comportamental, **los entornos dirigidos por eventos** encabezan el segundo tipo de EV3D revisado aquí y cuyo principal representante será el término acuñado como motor de juegos 3D (*game engine*). El comportamiento de un objeto 3D (cambios de posición, color, y demás propiedades) puede ser también definido en base a sus reacciones ante los cambios o eventos que ocurren en su entorno (p.ej: acciones del usuario, actuaciones de algún personaje inteligente, colisiones entre objetos, cambios de iluminación, etc.). Para ello, cada objeto interesado en la recepción de cualquier evento tendrá que incluir el código correspondiente a su manejo (*scripts*).

En este tipo de EV3D conducidos por eventos (*event driven*), todos los cambios son almacenados de forma global en una única cola y posteriormente encaminados hacia los objetos interesados en su recepción. Poseen, por tanto una naturaleza distribuida, adecuada para la integración de 3DIVA u otro tipo de modelo de razonamiento semántico asociado al entorno (ver punto 3).

Por otro lado, el modelo de comunicación global que ofrecen (frente al estático grafo de escena), se adaptará mejor a la simulación de mundos de naturaleza cambiante, proporcionando un método común para definir y atender estos cambios.

Veamos esto mediante un sencillo ejemplo de simulación: imaginemos que un avión (simulado) cae provocando una explosión de centro c y radio k unidades en un EV3D. Si cada unidad de terreno es capaz de modificar su estado, pero no posee información del resto del mundo, deberá consultar en cada ciclo si se ha producido algún cambio o colisión que le afecte (técnica de encuesta o *polling*). Por el contrario, si el EV3D posee un modelo de encaminamiento para los eventos que suceden en el mundo simulado, las unidades de terreno afectadas serán sencillamente avisadas al producirse un evento de colisión, modificando su comportamiento, es decir, animando la explosión.

Los ejemplos expuestos tanto a nivel geométrico (relación cuello \Leftrightarrow esqueleto) como en este nivel comportamental (explosión) sugieren la separación de ambos modelos. De esta forma se favorecerá el continuo desarrollo de ambos campos, íntimamente ligados al desarrollo de EV3D escalables y con posibilidades de reutilización en ambos niveles.

Distintos grupos de investigación han adaptado sus desarrollos a este tipo de entornos 3D como

plataforma de computación adecuada para la simulación de EVI3D, donde normalmente se integran distintos tipos de 3DIVA [9, 21, 6, 13, 8].

2.3. Interacción con el usuario

Una de las principales expectativas generadas en la mayoría de EV3D, es la libre interacción con los elementos 3D del entorno.

Los EV3D clásicos incluyen un modelo de interacción básico dentro de la jerarquía de nodos de su grafo de escena. Este modelo, centrado de nuevo en usuario, es representado por ejemplo en Java3D mediante los nodos *Viewpoint*, *NavigationalInfo* o *PhysicalEnvironment*, manteniendo información proveniente de algún dispositivo que registre posición, orientación del usuario y rutas de navegación del mismo. Esto es bastante útil a la hora del dibujado de la escena, ya que el grafo siempre poseerá información actualizada del usuario y la imagen será calculada desde su punto de vista. Por otro lado, resultará poco efectivo si el usuario quiere interactuar con una entidad situada fuera de su campo de visión. Por consiguiente, la inclusión del usuario en el grafo de escena limitará de nuevo el modo de interacción con el resto de nodos del grafo, es decir, con su entorno 3D.

Además, los nodos dedicados a la representación del usuario dentro del grafo escena carecen, en ocasiones, de la representatividad suficiente, omitiendo información, como la posición de ambos ojos (imagen estéreo), posición del torso, manos, etc. Por ejemplo, en VRML, el avatar o representante virtual del usuario en la escena 3D, está únicamente representado por una altura, una anchura (radio de colisión), y un desplazamiento máximo para sus pasos. EV3D más recientes como el SVE [14], incluyen cada una de las partes geométricas importantes del usuario (manos, ojos, cabeza, etc).

3. Entornos Virtuales Inteligentes

El énfasis de los Entornos Virtuales Inteligentes (EVI3D) [17] reside en incrementar las capacidades comportamentales e interactivas de los EV3D. Esto se consigue mediante la incorporación de capas o sistemas de IA situados e interactuando con

el sistema gráfico.

Aunque existen diferentes técnicas de animación (p.ej: interpoladores VRML o nodos de secuencia de OpenGL-Performer) y sencillos modelos comportamentales que pueden ayudar a dotar de cierto dinamismo a los entornos clásicos, en la mayoría de ocasiones, su naturaleza de guión o plan precompilado (*off-line*) sólo logrará mantener el interés del usuario durante un tiempo bastante limitado.

Un paso adelante en este sentido es, por ejemplo, la implementación en VRML y VRCC (un lenguaje de programación sincronizado) de criaturas 3D inspiradas en sus comportamientos biológicos [7]. Sin embargo, este enfoque se ve limitado, entre otras cosas, por el modelo comportamental de VRML, que requiere la declaración explícita de las rutas de eventos que entran y salen del grafo de escena. Esto en la practica se traduce en la imposibilidad de implementar comportamientos complejos (basados en motores de razonamiento) en el EV3D, debido a que VRML no dispone de mecanismos internos que faciliten la implementación y el control de eventos asíncronos.

A continuación revisamos tres formas de incrementar las capacidades interactivas de un EV3D, mediante la integración de sistemas de IA, y explicamos, como los EV3D basados en la distribución asíncrona de eventos favorecen esta integración.

- **Incremento de la interactividad del EV:** Una forma incrementar la interactividad en el EV3D, es mediante la integración de motores de razonamiento simbólico que interpreten, por ejemplo, las acciones del usuario. Un ejemplo es el sistema de configuración inteligente desarrollado en [5], donde el usuario puede navegar libremente en el EV3D e interactuar con el problema de configuración espacial definido. El sistema, es sensible a estos cambios de entorno, siendo capaz de recolocar automáticamente los elementos necesarios, de forma que se obtenga una posible solución a distintos problemas de diseño y/o configuración del mismo.

La integración entre el motor gráfico (UnrealTM) y el motor de razonamiento (desarrollado en GNUProlog) es alcanzada a través de un modelo de eventos que funciona en ambas direcciones y que se beneficia de los mecanismos incorporados al motor gráfico para tratar con eventos asíncro-

nos. Por ejemplo, supongamos que el usuario quiere modificar la configuración propuesta por el sistema y decide colocar un objeto en otra posición en el EV3D. Esta interacción del usuario es encapsulada en un evento generado por el entorno gráfico y es pasada a través al motor de razonamiento quien genera una nueva configuración, en la forma de un mensaje de texto (figura 5). La recepción de éste genera una serie de eventos de bajo nivel (p.ej: *spawn*, *receivedline*, etc) en el motor gráfico que hacen posible que la nueva configuración sea visible al usuario. Este es un modelo general que puede aplicarse a varias operaciones en los objetos: cambio de propiedades del objeto, estado, etc.

Siendo lo anterior muy significativo, el aspecto más importante es que los motores de razonamiento se pueden integrar a la perfección en motores gráficos si éstos están basados en eventos asíncronos. Es decir, determinados eventos, como por ejemplo *drop*, tienen que ser capturados para su **procesamiento** en el motor simbólico pero otros no. Así, una aplicación puede estar interesada en capturar las acciones del usuario con objetos: coger, dejar, empujar, chocar, etc, pero no en como los objetos se comportan desde el punto de vista físico: como caen, como ruedan, etc. Estos son gobernados y controlados por las mecanismos internos del motor gráfico.

- **Necesidad de una representación del conocimiento:**

La integración de personajes inteligentes 3D en estos entornos virtuales ejemplifica la necesidad de una representación del conocimiento y los requerimientos que esto conlleva.

Los 3DIVA deben manejar información a distintos niveles: el nivel bajo típicamente recogerá la información geométrica (listas de vértices, esferas o cajas envolventes, etc.), accesible en todo momento para cualquier agente de simulación y el nivel alto (semántico) permitirá a los agentes resolver distintos problemas en el mismo. De cara a poder representar y manejar este nivel semántico, los 3DIVA deben cumplir dos importantes propiedades: en primer lugar, poseer algún lenguaje de acción con el que representar los cambios que producen en el EV3D (ver extensiones de STRIPS en la

figura 3, y en segundo lugar, disponer de algún mecanismo de comunicación que les permita invocar la ejecución de sus acciones motoras en el EV3D, modificando así el estado del mundo en caso de poder llevarlas a cabo.

<pre> Operator: Give_Gift Preconditions: Gift true See_Rachel true Rachel_Free true Offer_Gift false Effects: Success: Gift false Hands_Empty true Affection +4 Fail: Offer_Gift true Rachel_Free false Affection -2 Mood -3 Actions: 2 Go_To_Rachel Send_Msg_Offer_Gift </pre>	<pre> Operator: Say_Nice_Things_To_Her Preconditions: See_Rachel true Rachel_Free true Be_Friendly false Mood > 6 Shyness < 6 Effects: Success: Be_Friendly true Rachel_Free false Affection +5 Mood +3 Fail: Rachel_Free false Affection -1 Mood -2 Actions: 2 Go_To_Rachel Send_Message_Nice_Things </pre>
--	--

Figura 3. Operadores utilizados en story-telling [16]

El mantenimiento del **modelo semántico** asociado al EV3D es un paso necesario para la integración de distintos agentes *situados* en un mundo compartido. De esta forma se facilita la integración de mecanismos de cooperación entre 3DIVA, tal y como muestra la figura 6 donde varios agentes cooperan en el conocido mundo de bloques, gracias a la percepción del estado simbólico en su campo de visión [8]. Para que esta cooperación pueda producirse es necesario disponer de un modelo semántico (dinámico) que muestre en todo momento el estado visible de los objetos del EV3D y maneje sus cambios. De esta forma, los cambios producidos por un agente (p.ej: coger/dejar cubo) son percibidos por el resto de agentes, pudiendo adaptar sus planes en caso necesario.

Por consiguiente, los entornos basados en eventos encajan perfectamente en este tipo de protocolos de acción, donde típicamente los agentes envían peticiones de actuación al entorno3D y esperan el *feedback* asociado. Además, estos eventos pueden encapsular la información semántica necesaria (átomos) para la actualización de la memoria del agente, el cual podrá de esta forma adaptarse al mundo cambiante que percibe.

Por ejemplo, en [16, 18] se compararon varios formalismos de planificación adecuados a actores virtuales. Los operadores basados en STRIPS y utilizados en este problema típico de *story-telling* se muestran en la figura 3. En este caso, se permite cierto grado

de no determinismo en la actuación, por lo que las acciones, además de fallo, pueden devolver distintos resultados, que serán encapsulados en sus correspondientes eventos y enviados al agente tal y como ya hemos comentado.

- Alternativa a la simulación física:** Un ejemplo en este sentido es la idea de la Realidad Alternativa (RA) como una nueva forma de experimentar entornos virtuales. Esto es, RA se define como la experiencia producida por entornos virtuales cuyo comportamiento parte de las experiencias *normales* en el mundo real [4]. Por ejemplo, el comportamiento de los objetos está determinado por las leyes físicas que establecen principios de dependencia y causalidad. En RA, estos principios son manipulados y el resultado es una realidad *alternativa*. Por ejemplo, Cavazza et al han experimentado con distintas técnicas para interceptar y manipular los *eventos* que rigen los fenómenos físicos en un entorno virtual. Por consiguiente, como en los casos anteriores, los mecanismos incorporados a los entornos virtuales basados en eventos asíncronos facilitan la incorporación de estas técnicas.

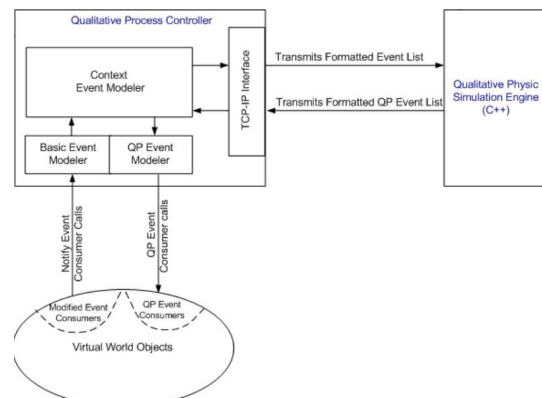


Figura 4. Arquitectura para el control y manipulación de eventos en RA [4]

Un ejemplo de la arquitectura de este tipo de sistemas es ilustrado en la figura 4. La figura muestra como eventos específicos, en este caso relacionados a procesos físicos como el flujo del agua, son interceptados, manipulados y enviados a un motor de simulación, en este caso de simulación cualitativa de procesos físicos. Como en los casos anteriores, los demás eventos son gobernados y controlados por los mecanismos internos del

motor gráfico. Así, es fácil ver como este tipo de tecnologías facilita la implementación de esta nueva generación de EV3D: entornos virtuales 3D inteligentes.

4. Conclusiones

La evolución natural de los EV3D, pasa por proporcionar entornos de simulación adecuados a mundos 3D de naturaleza cambiante, y en este sentido, los entornos dirigidos por eventos proporcionan actualmente un marco adecuado para la investigación y desarrollo en EVI3D.

Los modelos comportamentales y de comunicación clásicos, no satisfacen suficientemente estos requerimientos, debido principalmente a su alta dependencia con el modelo geométrico (grafos de escena), el cual está claramente orientado a una rápida visualización centrada en el usuario, dejando de lado importantes cuestiones comportamentales y de interacción.

Finalmente, la integración de EVI3D, es un problema raramente tratado desde el punto de vista del modelo computacional que soporta el EV3D. Dado que estos entornos están llamados ser los ambientes de simulación de distintos tipos de EVI3D, deberán ser capaces de gestionar los nuevos requerimientos dinámicos impuestos. En este sentido, los modelos basados en el envío de recepción asíncrona de eventos proporcionan un modelo de simulación adecuado para la actual intersección entre la IA y los Gráficos 3D en tiempo real.

5. Agradecimientos

Los autores queremos agradecer el interés y buenas ideas de Rafael Rodríguez (Brainstorm.es) en la elaboración de este artículo.

Referencias

- [1] VRML Consortium: VRML97 International Standard (ISO/IEC 14772-1:1997).
- [2] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. *ACM Transactions on Graphics*

hics (ACM SIGGRAPH 2002), 21(3):612–619, 2002.

- [3] Avi Bar-Zeev. Scenegraphs: Past, present and future. <http://www.realityprime.com/scenegraph.php>.
- [4] Hartley S. Lugin J.L. Cavazza, M. Technical report: Alternative reality technologies. , *Deliverable No D-5.0 of the ALTERNE project (IST-38575)*, Unpublished, 2003.
- [5] Calderon C. Cavazza M., Diaz D. Interactive problem solving in an intelligent virtual environment. *Proc of the 2003 international conference on Intelligent user interfaces (ACM). Florida(USA)*, pages pp 319 – 319, 2003.
- [6] Cavazza M. Charles F., Mead S.J. Character based interactive storytelling. *IEEE Intelligent systems*, July/August 2002, 17-24.
- [7] P. Codognet. Animating autonomous agents in shared virtual worlds. *Proceedings DMS'99, IEEE International Conference on Distributed Multimedia Systems*, Aizu Japan, IEEE Press 1999.
- [8] M. Lozano et al. An efficient synthetic vision system for 3d multi-character systems. *Lecture Notes in Computer Science (Intelligent Agents, 4th International Workshop, IVA 2003)*, 2003.
- [9] Steve Schaffer Chris Sollitto Rogelio Adobati Andrew N. Marshall Gal A. Kaminka, Manuela M. Veloso. Gamebots: A flexible test bed for multiagent research. *Communications of the ACM*, January 2002:43–45, 45 No 1.
- [10] <http://oss.sgi.com/projects/inventor/>.
- [11] <http://www.j3d.org/>.
- [12] <http://www.sgi.com/software/performer/>.
- [13] John C. Duchi John E. Laird. Creating human-like synthetic characters with multiple skill levels: A case study using the soar quakebot. *AAAI 2000 Fall Symposium Series: Simulating Human Agents*, November 2000.
- [14] G. Drew Kessler. Virtual environments models. *In Handbook of Virtual Environments K Stanney, Ed., Lawrence Erlbaum Associates*, 2002, 255-276.

- [15] Sbretchs M Clawson D Higgins G. Lathan C.E, Tracey M. Using virtual environments as training simulators: Measuring transfer. *In Handbook of Virtual Environments K Stanney, Ed., Lawrence Erlbaum Associates., 2002, 403-415.*
- [16] Charles F. Lozano M., Mead S.J. Bisquerra A. Cavazza M. Planning formalisms and authoring in interactive storytelling. *Proc on 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment, March, 2003 Darmstadt Germany.*
- [17] Aylett R. Luck M. Applying artificial intelligence to virtual reality: Intelligent virtual environments. *Applied Artificial Intelligence, 2000.*
- [18] A. Bisquerra M. Lozano, F. Barber. Formalismos de planificación para la animación comportamental de humanos virtuales. *XIII Congreso Español de Informática Gráfica (CEIG2003), A Coruña 2-4 Julio del 2003.*
- [19] R. Bindiganavale. N. Badler, M. Palmer. Animation control for real-time virtual humans. *Communications of the ACM, 42(8):64-73, August 1999.*
- [20] L.Moccozet G.Sannier A.Aubel P.Kalra, N.Magnenat-Thalmann. Real-time animation of realistic virtual humans. *IEEE Computer Graphics and Applications, 1998:42-55, 18 No5.*
- [21] Young R.M. An overview of the mimesis architecture: Integrating intelligent narrative control into an existing gaming environment. *Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment AAAI Press, 2001.*
- [22] Seron F.J. Rodriguez R., et al. Adding support for high-level skeletal animation. *IEEE Transactions on Visualization and Computer Graphics, pages 360-372, 2002.*

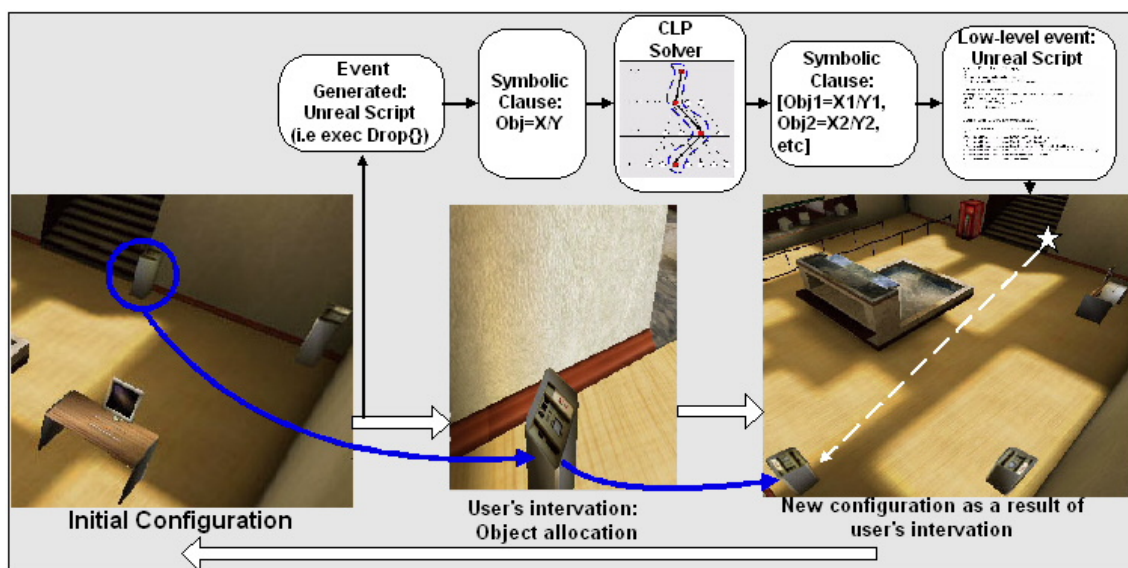


Figura 5. Modelo de interacción, A: configuración es propuesta por el sistema, B: el usuario recoloca un objeto, C: el sistema responde mediante una nueva configuración acorde a las restricciones.

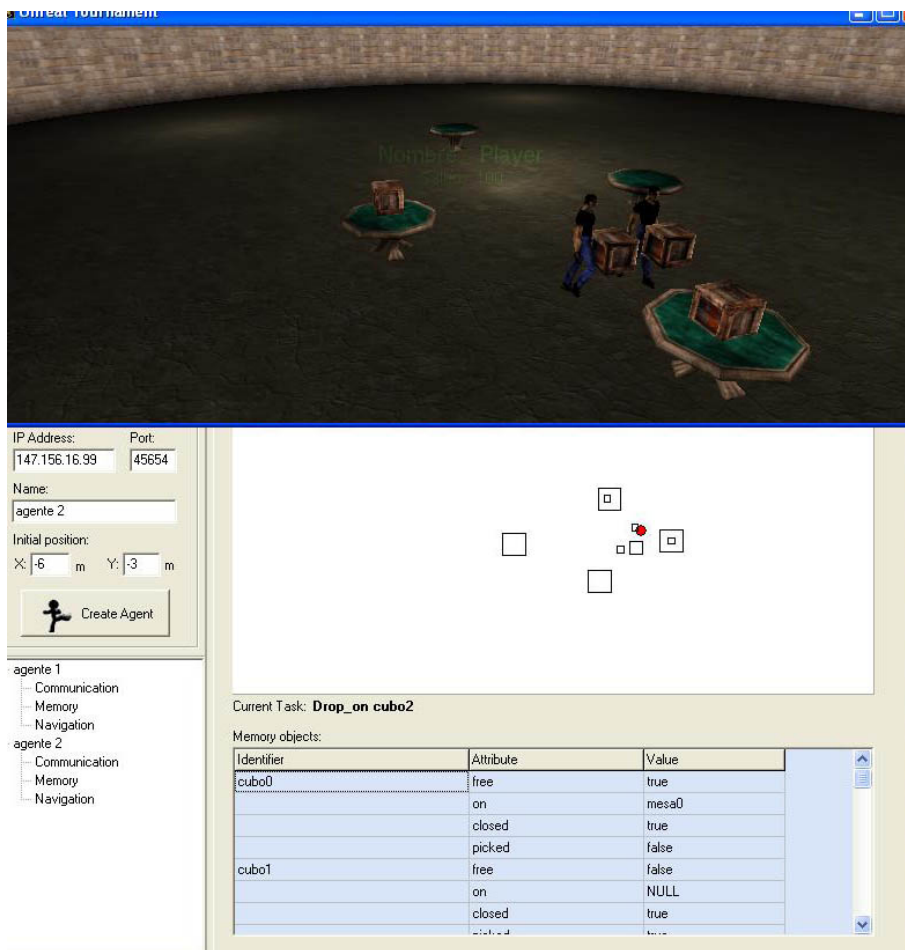


Figura 6. 3DIVA cooperando en un mundo compartido.