Universidad de las Ciencias Informáticas

Facultad 5

Sistemas de animación para la SceneToolKit

Informe del Aporte Personal al Proyecto Herramientas de Desarrollo para Sistemas de Realidad Virtual

Presentado en opción al título de

Máster en Informática Aplicada

Autor: Ing. Fernando Jiménez López

Tutor: MsC. Yuniesky Coca Bergolla

Ciudad de la Habana

Junio de 2009

Declaración jurada de autoría

Yo, Fernando Jiménez López, con carné de identidad 80041209146, declaro que soy el autor principal del resultado que expongo en la presente memoria titulada "Sistemas de animación para la SceneToolKit", para optar por el título de Máster en Informática Aplicada.

Este trabajo fue desarrollado durante el período 2003–2009 en colaboración con mis colegas de equipo quienes me reconocen la autoría principal del resultado expuesto en esta memoria.

Autorizo a la UCI a hacer el uso que estimen pertinente de los resultados aquí presentados como propietaria de los derechos legales de este proyecto.

Declaro que todo lo anteriormente expuesto se ajusta a la verdad, y asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de la Habana a los 10 días del mes de junio del año 2009.

Fernando Jiménez López

Agradecimientos

A Leticia y a Fernandito, por tener que compartir su tiempo con mi trabajo.

A Coca y a Mayra, sin ellos este trabajo no hubiera sido posible.

A todo el equipo de trabajo que ha transitado por el proyecto y ha puesto su grano de arena en la construcción del mismo.

A todo el que nos ha apoyado.

Síntesis

Enfrentar proyectos con tecnologías de Realidad Virtual (RV) ha sido un reto para el devenido Polo Productivo de Realidad Virtual de la Facultad 5, en la Universidad de las Ciencias Informáticas (UCI). Este entorno propició el surgimiento del proyecto "Herramientas de desarrollo para Sistemas de Realidad Virtual" (HDSRV). Su desarrollo ha estado acompañado por un conjunto de experiencias y resultados. El más significativo es la creación de un paquete de herramientas, bautizado **SceneToolKit** para el desarrollo de sistemas de realidad virtual.

En el inicio de este documento se hace una síntesis tanto del proyecto en general como de su principal producto, a través de un conjunto de valoraciones sobre sus logros y deficiencias. Se toma como base el Informe Final de Proyecto para esta etapa y las experiencias alcanzadas por el autor como miembro del equipo de desarrollo.

Luego, el trabajo se centra en torno a los sistemas de animación y las soluciones arquitectónicas resultantes del aporte intelectual del autor del presente documento, al mencionado proyecto.

Palabras Clave: Videojuegos, motor gráfico, motor de juego, realidad virtual, simuladores, *graphic engine*, *game engine*, animación.

Índice de contenidos

1. Introducción

Los Sistemas de Realidad Virtual (SRV) gozan de gran popularidad en la actualidad. Se han expandido considerablemente a diferentes sectores de la sociedad en los últimos años, pues esta tecnología ofrece prestaciones de una manera intuitiva, segura y económica. Puede encontrarse en aplicaciones para la industria, la defensa, la medicina, la educación, el entretenimiento, entre otras.

La Facultad 5 de la Universidad de las Ciencias Informáticas (UCI) orientó su especialización hacia la investigación y el desarrollo de productos usando esta tecnología. El primer proyecto de envergadura que enfrentó —en el año 2004— fue la construción de un simulador de conducción de vehículos automotores en colaboración con la empresa SIMPRO[1].

El desarrollo de toda la tecnología de Realidad Virtual (RV) se apoya arquitectónicamente en un núcleo central —también denominado "motor de simulación"— que se encarga de accionar los diferentes módulos para el funcionamiento del sistema. Existen muchas propuestas de estos motores a nivel mundial e incluso algunas bajo licencias libres, sin embargo, la dirección de producción de la Facultad 5 con la asesoría de los especialistas de SIMPRO, decidió el desarrollo de un motor propio y un conjunto de herramientas auxiliares para dar soporte a la creación de SRV. Esto dio lugar al proyecto "Herramientas de Desarrollo para Sistemas de Realidad Virtual" (HDSRV). La decisión se fundamentó de manera general en la necesidad inicial de tener un sistema

[1]SIMPRO, Empresa de SIMuladores PROfesionales: alias comercial del Centro de Investigaciones y Desarrollo #2 (CID-2) de la Unión de Industrias Militares (UIM) de las FAR.

base confiable y a la medida, que garantice la soberanía tecnológica y con la ventaja de actualizaciones futuras que se integren a partir de la arquitectura base, evitando pérdida de rendimiento por incompatibilidades entre módulos y que se nutra de las investigaciones que surjan a través del desarrollo de los productos. Más detalles se pueden encontrar en el Informe del proyecto [CJ09], en los epígrafes 1.2.1 y 1.2.2, y los epígrafes 1.6.1 y 1.6.2.

Una de las prestaciones de mayor importancia —y requerida por todas las aplicaciones que hacen uso de estas herramientas— es la representación de elementos animados en los entornos virtuales. Las animaciones son un componente fundamental para cualquier escenario si se busca una aproximación a la realidad. La presencia de autos, personas, animales, el cambio de luces de un semáforo, y hasta el movimiento de las nubes son importantes para aumentar la inmersión en un SRV. La necesidad de contar con estas prestaciones generó el desarrollo de una investigación sobre este tema.

Como **problema** a resolver se definió: la carencia de funcionalidades para la generación de animaciones en el motor de simulación y videojuegos del Polo Productivo de Realidad Virtual, enmarcando su **objeto de estudio** al proceso de desarrollo de motores de simulación y videojuegos en el Polo Productivo de Realidad Virtual.

Se planteó como **objetivo general** desarrollar una colección de funcionalidades en torno a la generación de animaciones para el motor de simulación y videojuegos del Polo Productivo de Realidad Virtual, que responda a las necesidades de los proyectos y se integre de forma coherente a la arquitectura de dicho motor.

Se ubica el **campo de acción** en torno al proceso de animación de elementos en escenarios virtuales.

Para alcanzar este objetivo se planificaron un conjunto de tareas que se enuncian a continuación:

1. Constatar los logros y limitaciones existentes en el tema de las animaciones para entornos tridimensionales en tiempo real.
2. Determinar las diferentes formas de animación a desarrollar.
3. Definir las estructuras de datos, técnicas y algoritmos necesarios para implementar las propuestas.
4. Estructurar las funcionalidades de manera coherente a la arquitectura base del motor, garantizando bajo acoplamiento y alta cohesión.
5. Proponer la arquitectura base para soportar las herramientas externas de apoyo a estas funcionalidades y otras a desarrollar.

Luego del cumplimiento de las tareas de investigación planteadas y con la aplicación de los resultados alcanzados, se esperaba:

- Prestaciones para desarrollar diferentes tipos de animaciones, que presenten los requerimientos para una aplicación de tiempo real (*real time*).
- Permir la integración de las nuevas funcionalidades con la arquitectura base definida en el motor de simulación manteniendo el bajo acoplamiento y la alta cohesión.

- Crear una arquitectura y diseño base para herramientas complementarias a las funcionalidades añadidas, que permita su reutilización en la construcción de otras herramientas de su tipo.

Como resultado general del proyecto HDSRV, se obtuvo un paquete de herramientas nombrado **SceneToolKit** (**STK**), y su componente principal es un *engine* para simulación y videojuegos. Este se encuentra en la versión 8.06 ("Newton"), que incluye por primera vez un módulo completo de física.

SceneToolKit se ha nutrido del trabajo investigativo y de desarrollo de un colectivo de estudiantes y profesores del Polo, y presenta hasta el momento varios resultados en el desarrollo de simuladores, videojuegos y otras soluciones. Una visión completa de este paquete de herramientas puede encontrarse en los capítulos 2 y 3 del Informe de proyecto [CJ09].

Con respecto a las funcionalidades de animación específicamente —que es el aporte principal del autor de este trabajo— se incluyó en el motor un módulo para animación por jerarquías articuladas. Este permite obtener deformaciones suavizadas de mallas y el manejo de múltiples animaciones sobre un mismo modelo. Se incluyó además un sistema para animación por secuencia o desplazamiento de texturas y algunas funcionalidades para permitir el seguimiento de un elemento a través de trayectorias. Como complemento de esta última prestación se desarrolló una propuesta de herramienta para la construcción de grafos de trayectorias.

Otro de los aportes presentados en este trabajo —como resultado del desarrollo del paquete **STK**— es la propuesta de una arquitectura base para desarrollar una *suite* de herramientas con interfaz gráfica para facilitar la creación de SRV

basados en este motor.

La estructura del documento consta de dos momentos, en una primera parte se hacen una serie de valoraciones del autor acerca del proyecto HDSRV y el producto **STK**. Seguidamente se puntualiza sobre los aportes del autor del presente documento al desarrollo de esta solución. Como referencia principal se toma el Informe Final del Proyecto HDSRV [CJ09], disponible como anexo de este trabajo.

2. Desarrollo

A continuación se harán algunas valoraciones sobre el desarrollo del proyecto HDSRV y el paquete de herramientas **SceneToolKit**, presentados en el Informe Final de Proyecto [CJ09]. También se expondrá una síntesis de la contribución hecha por el autor de este trabajo a los resultados del proyecto.

2.1. Breve descripción del proyecto HDSRV

La concepción de la **STK**, como se expresa en el Informe Final de proyecto, parte de la necesidad de crear simuladores de entrenamiento en entornos de Realidad Virtual en menos tiempo y con la calidad requerida, donde los desarrolladores no tengan que profundizar demasiado en las complejidades gráficas concernientes a este tipo de sistemas.

En el primer capítulo del Informe se describe el proyecto y —a grandes rasgos— la evolución desde su surgimiento hasta la actualidad. Se exponen un conjunto de dificultades naturales en el período de consolidación del trabajo en equipo, donde la gestión de proyecto, la gestión de configuración, así como las metodologías de investigación y desarrollo han ido madurando con el tiempo, y a la par del sistema de producción de la Universidad. En la medida que se tornaron más consistentes los cuatro elementos planteados para el desarrollo de software (personas, proyecto, producto y proceso) [JBR04] —según el Proceso Unificado de Desarrollo de Software (RUP)— se fueron obteniendo mejores resultados.

2.1.1. El producto SceneToolKit

El producto resultante en este proceso se vio afectado por las dificultades antes mencionadas, lo que trajo consigo demoras para el desarrollo de las funcionalidades de forma independiente y de la integración de manera general. La incorrecta preparación del proceso de pruebas fue otro factor que determinó que no se lograran versiones estables en menor tiempo. Algunos de estos elementos se encuentran en el Informe Final de Proyecto página 44, en "Problemas detectados".

A pesar de ello el sistema fue evolucionando y perfeccionándose, tornando más sólida su arquitectura y definiendo con más precisión su estructura modular (ver resultado final en el epígrafe 2.4.2 "Arquitectura y módulos", del Informe señalado). Esto es debido a que se fue transitando, de un trabajo investigativo incipiente, hacia la formalización de investigaciones en las diferentes temáticas. Se propiciaron procesos para el desarrollo de tesis de grado, publicaciones, ponencias en eventos, entre otros resultados que fortalecieron el sistema de asimilación y generación del conocimiento. En el Informe Final de Proyecto epígrafe 1.5.1. "Investigaciones y publicaciones", se pueden encontrar los más relevantes.

Gracias a que actualmente este sistema presenta una estructura flexible y propicia para el cambio se puede considerar acertada la continuación de su desarrollo, a pesar de no presentar por el momento algunas prestaciones existentes en otros sistemas de su tipo en la actualidad. Es de señalar que, en función de las necesidades propias del Polo, tiene un nivel creciente de adecuación.

Hay varios puntos hacia los que debe orientarse el trabajo próximo, para dar

respuesta a las necesidades actuales de los proyectos que hacen uso de estas herramientas:

- Carencia de estructuras de datos espaciales clásicas como el Quadtree, Octree y BSPtree, existentes de manera general en todos los productos similares [Ebe01]. Estas estructuras de datos son perfectamente soportadas sobre las clases definidas actualmente para el grafo de escena, pero se dejó la implementación en manos de los proyectos que la requieran, pues existían otras prioridades para el desarrollo que cubrían a todos los miembros del equipo.

- El sistema de *render* actual está planteado de forma muy rígida y pobre. Basa el dibujado de todas sus geometrías en *Vertex Array*, especificado en la documentación de OpenGL 1.1 [Bly97], lo cual mejora el comportamiento con respecto al modo inmediato de *render* (*inmediate mode*); pero deben aparecer nuevas opciones de *render* como el *Display List*, existente en la misma especificación y el *Vertex Buffer Object* (VBO), provisto en las extensiones [Ope09] y soportado a partir de la versión de OpenGL 1.4. Luego de tener múltiples opciones de *render*, una misma escena constará de diferentes vías para el dibujado de los objetos geométricos, lo cual —usado de forma acertada— permite alcanzar buenas mejoras de rendimiento en el sistema. Esta estrategia ya está puesta en marcha mediante el estudio, adecuación e implementación de un sistema de *render* genérico, comentado en el epígrafe 3.2.2. “Render general” del Informe Final de Proyecto.

- El procesamiento de los estados de *render* pierde rendimiento por el uso innecesario de los cambios de estado. Esto es perfectamente mejorable

partiendo de la propia estructura del grafo de escena, que permite establecer estados de *render* para subárboles completos, evitando así un costo adicional. Esta dificultad también se gestionará desde el sistema de *render* mencionado en el punto anterior.

- No dar la posibilidad de incluir la programación de *shaders* es una de las ausencias más significativas de este *engine*. A pesar de que se ha tratado de incluir en versiones no estables con GLSL esto no ha llegado a establecerse definitivamente. La existencia de *pixel shaders* y *vertex shaders* con sus lenguajes correspondientes —que están especificados formalmente en el núcleo de OpenGl 2.0 [SA04] y DirectX 8 con HLSL [Mic09b]— es fundamental para dar un salto en la calidad de los productos que se desarrollen con este *engine*. Además, en función de las prestaciones de hardware existentes, profundizar en temas novedosos como la programación usando *geometry shaders*, que hizo su aparición con DirectX 10 [Mic09a]. Entre las primeras acciones planificadas en el desarrollo de este *engine* —como se ve en el epígrafe 1.6.2. "Nuevas estrategias" del Informe Final de Proyecto— está satisfacer esta necesidad.

- Crear herramientas complementarias que apoyen al *engine* y le facilite el trabajo a los desarrolladores, como editores de entornos, editores de sistemas de partículas, editores de componentes de interface o editores de personajes. Entre los resultados de este trabajo (epígrafe 2.2.4) se presentan las bases para crear este tipo de herramientas. Todo el trabajo realizado acerca de este tema se puede encontrar en los epígrafes de la sección 3.5 del Informe Final de Proyecto.

Hasta el momento, y con las prestaciones existentes, se aprecian algunos re-

sultados relevantes con el uso del paquete **STK**, que pueden encontrarse en la sección 2.7. “Resultados positivos” del Informe Final de Proyecto.

2.2. Síntesis de la contribución personal

En esta sección se abordarán los temas que —como contribución intelectual— el autor del presente trabajo aportó al proyecto HDSRV y que se encuentran descritos en el Informe del mismo. En cada uno de ellos es común la integración de las estructuras creadas con la arquitectura base del *engine*, fundamentalmente en lo que concierne a los nodos del grafo de escena y los controladores. Esta integración, que garantiza el uso óptimo de las potencialidades arquitectónicas del sistema, es una de las fortalezas de los trabajos realizados.

2.2.1. Animaciones basadas en *Sprites*

Como ya se ha mencionado, los efectos de animación siempre son deseados en los escenarios virtuales, pero estos vienen acompañados de costos adicionales de procesamiento, que en muchos casos ponen en riesgo el comportamiento de tiempo real. Esto puede estar dado por la complejidad del procesamiento o por las restricciones mínimas de hardware que presentan los usuarios finales del sistema. Una solución aplicada para lograr el efecto visual deseado —sin pérdidas en el rendimiento de la aplicación— es sustituir costosos efectos por imágenes preprocesadas que representen los efectos deseados [AGAM03]. En este caso los artistas gráficos son los principales responsables de la calidad y veracidad del efecto según la ubicación en el contexto de la escena. En términos de programación, se requiere entonces lograr representar esas imágenes o secuencias de

imágenes con la opacidad, posición, orientación, velocidad de cambio u otras propiedades que haya definido el creador del efecto, sobre el tipo de geometría requerida. Todo esto queda abarcado por el sistema de animación basado en *sprites* desarrollado.

Estas prestaciones fueron adicionadas al *engine* y comentadas en el Informe Final de Proyecto, epígrafe 3.3.9. "Animaciones por *sprites*", donde se mencionan varios resultadas de la aplicación de este tipo de animación.

Los *sprites* por lo general —en las diferentes plataformas que brindan esta prestación, como SDL [Gam09] o DirectX [Mic09c]— parten de una imagen que contiene a espacios iguales los diferentes fotogramas de la animación. De esta manera se puede secuenciar la animación a través del cambio de las coordenadas del mapa de bit de la imagen. Para un *engine* que maneja escenarios 3D como es el que se está tratando, esto equivale a cambiar las coordenadas del mapa de textura.

Sin embargo, existía un contratiempo con el uso de esta técnica. Este consistía en que al equipo de diseño le era costoso en cuanto a tiempo preparar una textura donde quedaran ubicadas con presición las imágenes, de manera tal que con el paso de las secuencias se mantubieran centradas. Entonces se añadió la posibilidad de adicionar la secuencia a través de imágenes separadas, lo cual mejoró grandemente la velocidad de las entregas por parte del equipo de diseño.

Inicialmente el propósito fue representar las imágenes sobre un plano de dimensiones variables, con lo que se lograron casi todos los resultados comentados en el Informe. Pero la idea se extendió hacia el uso de cualquier tipo de geometría, lo cual se resume a generalizar algunas funcionalidades en las clases

manejadoras de *sprites*.

Con esta técnica se pueden alcanzar nuevas prestaciones como representar la animación del rostro de un modelo humano, sin requerir la deformación de la geometría. Esto resulta muy beneficioso cuando se aplica a elementos situados en un segundo plano de la escena virtual, y es de sumo interés para el desarrollo de proyectos de simulación tanto de conducción (para los peatones) como de tiro (en soldados enemigos), sin comentar su aplicación en videojuegos.

Uno de los principales aportes de la técnica de *sprites* en este *engine* es que ha permitido llevar sistemas que recrean escenarios tridimensionales atractivos a PCs de bajas prestaciones gráficas, como son las existentes en las escuelas del Ministerio de Educación (MINED) en el país.

2.2.2. Animaciones por jerarquías articuladas

Como se plantea en el epígrafe 3.1.6. "Animaciones", del Informe Final de Proyecto, la necesidad de animar personajes, y especificamente a través de jerarquías articuladas, es uno de los requisitos iniciales solicitados desde el surgimiento del proyecto y fue implementado luego de existir las funciones básicas para el dibujado de la escena. El propósito radicaba en conseguir animaciones realistas, y esta propuesta es la idónea para asimilar la información que brindan los sistemas de captura de movimientos (*motion capture*) [Met09], generadores de animaciones de alta calidad y realismo.

El estudio inicial se presentó en la tesis de pregrado [CJ04] del presente autor. Aunque en ésta fueron abordados de manera somera otros temas de la ani-

mación, se utilizó como una de las referencias bibliográfica un artículo [Lan98] publicado por la revista "Game Development", y que ha sido contínua referencia en trabajos posteriores. Sobre esa base se desarrolló este trabajo.

Actualmente, este sistema de animación realiza el procesamiento en el CPU, debido a las limitaciones que presenta el sistema de *render* del *engine* — y que se comentaron en el epígrafe 2.1.1 de este documento— pero una considerable mejora de rendimiento puede lograrse pasando al GPU esta responsabilidad, por las ventajas respecto a la velocidad de cálculo. Esta optimización es una de las razones para no emplear una biblioteca libre de animación de personajes 3D llamada Cal3D [Cal06], usada en otros sistemas similares como Delta3D [DMJ05]. En esta biblioteca se abstrae completamente el uso de las API gráficas.

La integración de esta técnica con la arquitectura del gráfo de escena, presentada en el epígrafe 3.1.5. "Grafo de escena" del Informe Final de Proyecto, es casi de forma natural. La jerarquía de huesos funciona como una extensión de esta estructura. Sin embargo, se organizaron los huesos de forma secuencial en función del recorrido en preorden necesario, ya que esta estructura no cambia y así se evita el costo de usar métodos recursivos para lograr los mismos resultados. No obstante, es importante mantener en los huesos las características generales del grafo de escena para lograr adicionar, en determinadas circunstancias, elementos externos a esta jerarquía como puede ser —en un videojuego— el arma en la mano de un personaje.

Como un nivel más alto de este módulo, se crearon un conjunto de funcionalidades para la manipulación de animaciones en un personaje, tratado en el apartado "Manipulación de personajes" del epígrafe 3.4.1."Sistema dinámi-

co". Este es vital para facilitarle el trabajo de creación de un personaje a los desarrolladores de proyectos, donde se incluyen funcionalidades muy deseadas como la mezcla (*blending*) de animaciones y la transición entre las mismas.

Un elemento que debe añadirsele para completar estas prestaciones es la capacidad de crear un grafo de transición de animaciones, donde se estructuren las posibles variantes de paso de un estado a otro en el comportamiento dinámico de un personaje.

La información de las animaciones y de los pesos de los huesos se encuentran en ficheros ANX y PWX, como puede verse en el epígrafe 3.1.1. "Manipulación de ficheros".

2.2.3. Animaciones por seguimiento de trayectorias

En los epígrafes anteriores se presentaron dos técnicas para el desarrollo de animaciones aplicables a personajes u otros elementos de la escena. Pero estas animaciones deben ser lo suficientemente genéricas como para poder reutilizarse por todo el sistema. Una de las condiciones que deben cumplir es que no incluyan el desplazamiento, para luego poder combinar estas animaciones —que pueden ser de correr o saltar en el caso de un personaje— con el desplazamiento requerido. Entonces, para lograr un completo funcionamiento de personas, autos, animales u otros elementos en un escenario virtual, que requieran de trayectorias autónomas, se crearon inicialmente las prestaciones del apartado "Animación por seguimiento de trayectorias" del epígrafe 3.4.1."Sistema dinámico", en el Informe Final de Proyecto.

Una de las virtudes que tiene este sistema es que se le puede asociar (como controlador) a cualquier elemento o grupo de elementos presentes en el grafo de escena (ver epígrafe 3.1.5 del Informe Final de Proyecto), pues adquirió la flexibilidad de la clase CController. Con esta funcionalidad se animaron autos, personas y otros elementos en simuladores deserrollados. Haciendo una comparación con el *engine* Ogre3D [Ogr08], —propuesto como alternativa al uso de **STK** (ver sección 1.6 del Informe)— se puede encontrar una funcionalidad similar a esta, sin embargo sólo fue concebida para personajes, pues es de uso exclusivo para elementos con estructuras de huesos.

Como complemento se creó una herramienta que extendiera el seguimiento de caminos a niveles más complejos. En el epígrafe 3.5.3. "Herramienta de creación de grafos de caminos", del Informe Final de Proyecto, se presentan sus características. Básicamente, permite crear una red de caminos en un escenario dado. Entre las primeras aplicaciones de esta herramienta está montar el sistema de recorrido de vehículos y personajes autónomos en una ciudad.

Es sumamente útil este sistema para el paquete **SceneToolKit**, ya que permite crear las estructuras del grafo sobre un visualizador de la **STK** embebido a la herramienta, lo cual deja ajustar los recorridos según la topografía del terreno y luego salvar toda la información en un fichero.

Con esta herramienta se salvan las aristas entre nodos como polilíneas, conforme al funcionamiento del controlador de trayectorias antes mencionado. Como esas polilíneas fueron construidas a partir de curvas de aproximación NURBS[2] —a pesar de mantenerse la interpolación lineal— los caminos tendrán trayectorias suavizadas sin necesidad de cálculos adicionales en tiempo

[2]Non Uniform Rational B-Splines.

real.

Queda por desarrollar para que el *engine* brinde estas prestaciones en su totalidad, la combinación de esta información con un sistema de IA que maneje las variantes de recorridos según sea conveniente.

2.2.4. Propuesta de arquitectura para herramientas auxiliares

SceneToolKit, como se ha visto antes, es un paquete de herramientas cuyo centro gira en función de su *engine*. El resto de las herramientas son complementos para apoyar el trabajo de este motor, brindándole a los proyectos que lo usen facilidades para agilizar el desarrollo de los productos. Las herramientas, de manera general, deben presentar estructuras que permitan: facilidad de una interfaz gráfica con menú, panel de herramientas, panel de propiedades, y una vista previa para ver los resultados del trabajo. A partir de ello se definió una arquitectura común puesta en práctica en la herramienta vista en el epígrafe anterior de este documento ("Herramienta de creación de grafos de caminos") y en el epígrafe 3.5.2. "Herramienta de configuración de entornos", del Informe Final de Proyecto, donde se puntualiza con más detalle esta solución.

Una de las características principales que presentan estas herramientas es que son desarrolladas con base en Qt [Nok09] versión 4.0 y **STK** versión 2.4, ambas trabajan sobre Windows y GNU/Linux, lo cual permite migrarlas muy facilmente. Esto es uno de los problemas presentados con las herramientas auxiliares de Ogre3D [Ogr08], que por lo general están en Windows.

Una de las tareas más interesantes realizadas para lograr este trabajo fue la

integración de un visualizador de la STK a un compomente de Qt. Este acople resultó de una menera genérica, que permitirá en próximas versiones de la **STK** crear aplicaciones basadas en interfaces de Qt, o de esa misma forma, crear aplicaciones en Qt con visualización basada en la **STK**.

3. Conclusiones

El reto que constituye desarrollar un paquete de herramientas para la creación de sistemas de realidad virtual, impulsa a un arduo trabajo investigativo. Actualmente, las crecientes capacidades de animación han sido un factor fundamental en el impacto de este tipo de sistemas. **SceneToolKit** no es una exepción, por lo que una de las líneas de I+D del proyecto HDSRV ha sido el desarrollo de prestaciones de animación para este paquete.

Los proyectos del Polo Productivo de Realidad Virtual, clientes primarios de este producto, tenían un conjunto de requerimientos de animación. Esto fue determinante para definir la prioridad en las técnicas de animación a desarrollar.

La inclusión de todas las prestaciones para crear animaciones con la **STK**, vistas en este documento, aunque han dado solución a las necesidades actuales de los proyectos del Polo, son parte de un proceso contínuo de desarrollo. Por ello, el trabajo actual constituye la base para la investigación y adecuación de nuevas tecnologías de animación que harán crecer las potencialidades del paquete **SceneToolKit**.

El tema de la arquitectura, tanto en la base del *engine*, como para la adición de nuevas prestaciones, ha sido vital en el desarrollo de todo el sistema. Mantenerla estable, con la flexibilidad requerida y el profundo conocimiento sobre la misma ha permitido la evolución y continuidad de este proyecto.

A. Glosario de términos

Términos en inglés:

***Game engine*:** Es el sistema operativo de un videojuego, el núcleo. Se encarga de dirigir y coordinar cada uno de los aspectos tecnológicos ligados a él. Capta eventos de entrada, computa física, reproduce sonidos, simula IA, pinta, etc.

***Graphic engine*:** (motor gráfico) Módulo gráfico independiente lo bastante potente para poder tratar toda la información que hay en él, así como su visualización en tiempo real. Este concepto surge por la complejidad gráfica añadida a la hora de tratar con escenarios 3D.

***Motion capture*:** (captura de movimiento) Adquisición de información de animación de un elemento real para ser reproducida en un escenario digital.

***Real time*:** En gráficos por computadoras, las aplicaciones en tiempo real son aquellas que van generando los fotogramas a medida que son necesarios durante la propia corrida de la aplicación. El tiempo real se consigue cuando el tiempo de respuesta de la aplicación es lo suficientemente corto como para que la percepción del fenómeno se corresponda con fidelidad al sistema real.

***Render*:** (rendering) Crear en forma automática una imagen de acuerdo al modelo tridimensional que existe en el ordenador.

***Shaders*:** Conjunto de instrucciones gráficas destinadas para el acelerador

gráfico, estas instrucciones dan el aspecto final de un objeto. Los *shaders* determinan materiales, efectos, color, luz, sombra y etc.

***Sprites*:** (duende, duendecillo, hada) Mapa de bits dibujados en la pantalla de ordenador por hardware gráfico especializado sin cálculos adicionales de la CPU. A menudo son pequeños y parcialmente transparentes, dejándoles así asumir otras formas a la del rectángulo. En otra acepción, las animaciones por *sprites* son aquellas que hacen uso de imágenes (generalmente cuadros unidos en una única imagen o tirilla) que se muestran en una secuencia dando la sensación de animación, y sustituyen animaciones de objetos tridimensionales usando solamente un plano.

Términos en español:

Estados de *render*: Información de estados (texturas, transparencia, material...) utilizada por el *renderer* para representar las geometrías de la escena.

Grafo de escena: Estructura jerárquica donde se almacenan y organizan los objetos de la escena para el control de su información y determinación de la visibilidad, donde las hojas contienen los objetos dibujables de la escena.

Motor gráfico: Ver *graphic engine*.

Polo Productivo de Realidad Virtual: Estructura organizativa existente en la UCI, que agrupa a los proyectos dedicadas al desarrollo de sistemas de realidad virtual.

Realidad Virtual: Representación de escenas o imágenes de objetos producidos por un sistema informático, que da la sensación de su existencia real.

SceneToolKit: Alias del paquete de herramientas desarrollado por el proyecto, formada por *scene*, escena, y *toolkit*, "grupo de herramientas". El nombre completo es "Herramienta para Desarrollo de Sistemas de Realidad Virtual", y debe constar del *game engine* y otras herramientas utilitarias y de edición.

Simuladores: Un simulador es un aparato que permite la simulación de un sistema, reproduciendo su comportamiento. Los simuladores reproducen sensaciones que en realidad no están sucediendo.

Sistemas de Realidad Virtual: Aplicaciones informáticas basadas en técnicas de realidad virtual (ver Realidad Virtual).

Tiempo real: Ver *Real time*.

Videojuego: Programa informáticom normalmente asociado a un hardware específico, que recrea un ejercicio sometido a reglas, se debe lograr uno o varios objetivos, donde los jugadores pueden interactuar y tomar decisiones.

B. Glosario de abreviaturas

API: *Application Programming Interface*, Interfaz de programación de aplicaciones.

GPU: Unidad de Procesamiento Gráfico (*Graphics Processing Unit*).

HDSRV: Nombre del proyecto: Herramientas de Desarrollo para Sistemas de Realidad Virtual.

IA: Inteligencia Artificial.

RV: Realidad Virtual.

SIMPRO: Empresa de Simuladores Profesionales: alias comercial del Centro de Investigaciones y Desarrollo #2 (CID–2) de la Unión de Industrias Militares (UIM) de las FAR.

SO: Sistemas Operativos.

SRV: Sistemas de Realidad Virtual.

STK: SceneToolKit.

UCI: Universidad de las Ciencias Informáticas.

Referencias

[AGAM03] Ayucar, I., A. García-Alonso y L. Matey: *Aplicaciones avanzadas del uso de texturas precomputarizadas en entornos de simulación de tiempo real.* Revista Ingeniería Industrial, páginas 45–57, 2003, ISSN 0717-9103. http://www.ici.ubiobio.cl/revista/index.php?option=com_docman&task=doc_download&gid=26&&Itemid=3.

[Bly97] Blythe, D.: *The OpenGL Graphics System. Specification version 1.1.* Informe técnico, OpenGL.org, 1997. http://www.opengl.org/documentation/specs/version1.1/glspec1.1/node1.html.

[Cal06] Cal3D project: *Cal3D – 3D Character Animation Library.* http://home.gna.org/cal3d/, 2006.

[CJ04] Camacho Román, Y. y F. Jiménez López: *Biblioteca Gráfica Para Sistemas de Realidad Virtual.* http://10.5.2.200:5800/trac/hdsrv/browser/trunk/Expediente%20Proyecto/investigaciones/tesis%202003-2004, 2004. CUJAE.

[CJ09] Camacho Román, Y. y F. Jiménez López: *Informe Final del Proyecto Herramientas de Desarrollo para Sistemas de Realidad Virtual.* Informe técnico, Polo Productivo de Realidad Virtual Facultad 5, UCI, 2009. http://10.5.2.200:5800/svn/hdsrv/trunk/Expediente%20Proyecto/informe%20de%20proyecto/InformeFinalProyecto.pdf.

[DMJ05] Darken, R., P. McDowell y E. Johnson: *The Delta3D Open Source Game Engine*. IEEE Computer Graphics and Applications, 25(3):10–12, Mayo–Junio 2005. http://www2.computer.org/portal/web/csdl/abs/mags/cg/2005/03/g3010abs.htm.

[Ebe01] Eberly, D. H: *3D Game Engine Design: A Practical Approach to Real-time Computer Graphics*, capítulo 12 Spatial Sorting, páginas 411–426. Morgan Kaufmann, 2001.

[Gam09] Game Dev Geek: *Animating Sprites with SDL*. http://gamedevgeek.com/tutorials/animating-sprites-with-sdl/, 2009.

[JBR04] Jacobson, I., G. Booch y J. Rumbaugh: *El proceso unificado de desarrollo de software*, volumen I, capítulo 2, páginas 13–30. Editorial Félix Varela, 2004.

[Lan98] Lander, J.: *Skin Them Bones: Game Programming for the Web Generation*. Game Developer, páginas 11–16, Mayo 1998.

[Met09] MetaMotion: *Motion Capture - What is it?* http://www.metamotion.com/motion-capture/motion-capture.htm, 2009.

[Mic09a] Microsoft Corporation: *Direct3D 9 to Direct3D 10 Considerations (Direct3D 10)*. http://msdn.microsoft.com/en-us/library/bb509703(VS.85).aspx, 2009.

[Mic09b] Microsoft Corporation: *HLSL*. http://msdn.microsoft.com/en-us/library/bb509561(VS.85).aspx, 2009.

[Mic09c] Microsoft Corporation: *Sprite (Clase)*. http://msdn.microsoft.com/es-es/library/microsoft.windowsmobile.directx.direct3d.sprite.aspx, 2009.

[Nok09] Nokia Corporation: *Qt Software*. http://www.qtsoftware.com/, 2009.

[Ogr08] Ogre3D Community: *Documentation Architecture*. http://www.ogre3d.org/wiki/index.php/Documentation_Architecture, Octubre 2008.

[Ope09] OpenGL.org: *OpenGL Extension Registry*. Informe técnico, OpenGL.org, 2009. http://www.opengl.org/registry/.

[SA04] Segal, M. y K. Akeley: *The OpenGL Graphics System: A Specification (Version 2.0 - October 22, 2004)*. Informe técnico, OpenGL.org, 2004. http://www.opengl.org/documentation/specs/version2.0/glspec20.pdf.

Fecha de creación: 11 de junio de 2009

Generado con LaTeX