Universidad de las Ciencias Informáticas

Facultad 5

Generación de Entornos Virtuales en Tiempo Real para Videojuegos

Informe de los Aportes Personales al Proyecto Herramientas de Desarrollo para Sistemas de Realidad Virtual

Presentado en opción al título de

Máster en Informática Aplicada

Autor: Ing. Omar Correa Madrigal

Tutor: MSc. Yanoski Rogelio Camacho Román

Ciudad de la Habana

septiembre del 2010

1. Declaración de autoría

Yo, Omar Correa Madrigal, con carné de identidad 83071311069, declaro que soy el autor principal del resultado que expongo en la presente memoria titulada "Generación de Entornos Virtuales en tiempo Real Para Videojuegos", para optar por el título de Máster en Informática Aplicada.

Este trabajo fue desarrollado durante el período 2007–2010 en colaboración con mis colegas de equipo, quienes me reconocen la autoría principal del resultado expuesto en esta memoria.

Autorizo a la Universidad de las Cencias Informáticas a hacer el uso que estime pertinente de los resultados aquí presentados, como propietaria de los derechos legales de este proyecto.

Finalmente 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 _ días del mes de septiembre del año 2010.

Ing. Omar Correa Madrigal

Agradecimientos

A quienes han contribuido a mi formación y mis resultados profesionales como Ingeniero en Ciencias Informáticas.

A toda mi familia por el apoyo y la comprensión recibida

A mi novia por todo su amor

A mis amigos, a todos, gracias.

...

Síntesis

La Generación de Entornos Virtuales en Tiempo Real es una temática muy abordada en estos días. Presenta grandes potencialidades para reducir el tiempo de desarrollo de los productos de Realidad Virtual y para la creación de entornos muy extensos con una gran variedad de contenidos. Estas ventajas la han hecho muy popular en el desarrollo de videojuegos. En este trabajo, se expone la labor del autor en la generación de entornos infinitos para videojuegos. Para ello, se recogen los aportes realizados al motor gráfico SceneTollKit(STK), al cual se le ha provisto de un sistema de generación flexible y novedoso, capaz de extender el uso de la STK más allá de los simuladores.

Palabras Claves: Generación en Tiempo Real, motor gráfico, videojuegos,simuladores, Realidad Virtual, SceneTollKit

Índice de contenidos

2. Introducción

El mundo del gráfico por computadoras ha alcanzado un gran desarrollo, tal es así que se encuentra en su cuarta generación [MH02].El aumento de la capacidad de procesamiento y memoria de las tarjetas gráficas, y de las PC en sentido general, han dado pie a la creciente calidad visual de estos sistemas; proporcionalmente la complejidad de los entornos ha aumentado también. Esto último coloca en una posición desventajosa a los desarrolladores, ya que deben crear entornos cada vez más grandes y con más contenidos significativos, lo cual siempre se traduce en un mayor esfuerzo y por tanto más tiempo de desarrollo. Reducir el tiempo de desarrollo y lograr aceptables niveles de visualización e interactividad en sus aplicaciones, constituye el principal reto al cual se enfrentan. La Generación Automática de Entornos Virtuales representa un campo de trabajo que ha ayudado sustancialmente al logro de esta meta. Muchos sistemas como: simuladores, sistemas geoespaciales 3D y videojuegos se han favorecido con su aplicación. Algunas aplicaciones como: FlaySimulator [Inn08],GoogleEath3D [Com08c], SeconLife [Inc08] y World of Warcaft [Cor08a]dan fe de ello.

Desde el 2003 en la Universidad de las Ciencias Informáticas, específicamente en la Facultad 5, se viene trabajando en el desarrollo de aplicaciones de Realidad Virtual, lo cual ha permitido alcanzar una experticia en el desarrollo de sistemas complejos como simuladores de conducción, sistemas de visualización médica, paseos virtuales y videjuegos. En este sentido, herramientas como el Motor Gráfico SceneToolkit(STK) y otras herramientas de soporte expuestas en [CGCA09], han potenciado el desarrollo de estas aplicaciones y reducido además su tiempo de desarrollo. Actualmente se trabaja en proyectos de vi-

deojuegos, por ejemplo el Proyecto Juegos CNEURO[1] , el cual se especializa en la creación de *videojuegos terapéuticos*. Los videojuegos con este corte, que pretendan personalizar sus contenidos según el grado de avance del paciente en la terapia, deben recrear entornos complejos y variables, por lo general infinitos [Mad09]. El Motor Gráfico STK, base de desarrollo seleccionada, carecía de las potencialidades para recrear entornos infinitos en *tiempo real*, ante tal situación, se plateó el siguiente problema: **¿Cómo generar Entornos Virtuales Infinitos en Tiempo Real a partir del Motor Gráfico SceneToolKit ?**

Con vista a resolver este problema, se identificó que la solución podía estar en **Las Técnicas y Algoritmos de Generación de Entornos Virtuales**, y más específicamente, en **Las Técnicas y Algoritmos de Generación de Entornos Virtuales Infinitos aplicables a videojuegos**.El objetivo que se plateó fue etonces: **Crear un Módulo de Generación de Entornos Virtuales en Tiempo Real, que permita crear Entornos Virtuales Infinitos para videojuegos usando el Motor Gráfico SceneToolKit.**

Para darle cumplimiento al mismo se plantearon las siguientes tareas :

- Caracterización de las técnicas y algoritmos de generación de entornos virtuales en *tiempo real* .
- Análisis de las bibliotecas comerciales y no comerciales, que permiten la creación de esta clase de entornos.
- Desarrollo de la solución en función del estudio realizado.
- Validación de la respuesta en *tiempo real* del sistema obtenido.

[1]Ver información sobre este proyecto en
http://ucipedia.uci.cu/index.php/Proyectos_de_Relidad_Virtual

En el transcurso del trabajo se logró a partir de los Métodos: Análisis y Síntesis, y Deducción e Inducción, identificar un grupo de técnicas y algoritmos que fueron tomados como parte de la solución. Luego, haciendo uso del Método de Modelación, se desarrolló la propuesta de solución y se obtuvo además una clasificación de las técnicas de generación de entornos virtuales para una mejor compresión del tema,algo no muy especificado actualmente. Para culminar, se sometío a prueba la solución y a partir de los Métodos de Medición y Observación se constató el logro del objetivo trazado.

A modo de síntesis, el autor de esta memoría, luego de su experiencia con el motor STK, y conociendo sus deficiencias para la generación de entornos infinitos en *tiempo real*, le proveyó al mismo de un módulo flexible, extensible e inteligente, con potencialidades para generar por el momento entornos Espaciales y de Paisajes. En esta propuesta, se le incorporó una novedosa estrategia de generación de entornos virtuales llamada Losas Dinámicas (Dynamic Tiles), la cual representa uno de los principales aportes. También se le incorporó al módulo un fichero de configuración que permite la reconfiguración estructural del generador para obtener otros entornos. Todo el estudio y los aportes realizados son valorados en el cuerpo de la presente memoria individual y expuestos con mayor profundidad en [CGCA09].

3. Desarrollo

3.1. Proyecto Herramientas para el Desarrollo de Sistemas de Realidad Virtual. Motor Gráfico SceneToolKit.

El Proyecto"Herramientas de Desarrollo para Sistemas de Realidad Virtual" (HDSRV), como se describe en el Capítulo 1 del informe del proyecto [CGCA09], ha encontrado en su camino dificultades que se han ido superando con el tiempo, en la medida en que la Universidad ha adquirido experiencias en la gestión de proyectos informáticos. Actualmente, ha pasado a un esquema de proyecto en comunidad, siguiendo las tendencias internacionales en el desarrollo del software y dado al éxito de comunidades como OGRE [Com08a], referente en este modelo a partir del desarrollo de su motor gráfico OGRE3D [Com08b].

Aunque el desarrollo en comunidad requiere de más tiempo de organización en cuanto a las tecnologías de comunicación, las experiencias hasta el momento han permitido crear nuevas herramientas y módulos para el Motor Gráfico SceneToolKit(STK), lo cual ha elevado sustancialmente sus prestaciones, y todo, desde el trabajo en los proyectos productivos [CGCA09]. Actualmente, el uso de la STK se ha extendido a la creación de videojuegos[2] , algo para lo cual no se encontraba preparada. Esto permite identificar que el enfoque de la herramienta ha ido evolucionando de su concepción genérica a versiones más específicas según el tipo de producto. La organización en comunidad ha permitido fomentar la creación de Motores de Videojuegos como es el caso del

[2]Energía para Aprender y Metoerix constituyen los ejemplos más representativos. http://ucipedia.uci.cu/index.php/Proyectos_de_Relidad_Virtual

CNEUROGameEngine del Proyecto Juegos CNEURO. Todo indica que en un futuro no muy lejano la STK se extenderá a otros campos de aplicación, lo que elevará sustancialmente sus prestaciones y generará además un material científico de alta calidad.

3.2. Estado del Arte en la Generación de Entornos Virtuales.

El avance alcanzado por el hardware y la creciente necesidad de lograr visualizaciones más realistas, han permitido obtener a través del gráfico por computadoras resultados realmente impresionantes. Aunque los dispositivos de almacenamiento y de procesamiento permiten aspirar a niveles aceptables de visualización, lo cierto es que la complejidad de los sistemas sigue colocándolos en crisis, por lo cual continua latente la necesidad de la eficiencia en la construcción y representación de los Entornos Virtuales.

La curiosidad de un jugador o de un usuario de cualquier Sistema de Realidad Virtual, eleva constantemente la demanda de crear entornos más extensos y con menos restricciones de movimiento. Los desarrolladores, a pesar del avance de las tecnologías de diseño, presentan siempre problemas con el tiempo y el esfuerzo para elaborarlos. Por tal motivo la Generación Automática de Entornos ha devenido siempre como campo de apoyo para resolver tales problemas, permitiendo así, grandes niveles de desarrollo. Herramientas y algoritmos para toda clase de entornos han sido elaborados [GS04]. Gracias a estos el tiempo que puede tomar para crear un entorno de interiores de cualquier tamaño, puede ir desde 2 segundos hasta 6 minutos [Ada02], algo que contrasta con

la estadística: “la creación de un videojuego con 14 horas de contenidos relevantes puede tomar hasta 1 año de trabajo” [S.I99], si se analiza que la fase de creación de los contenidos de un juego es la que mayor tiempo consume. Por otra parte, la generación brinda otras potencialidades, como es la creación de entornos infinitos [GS04] [GPSG03] [Cor08a] en *tiempo real* con una gran variedad de contenidos, válidos para videojuegos y simuladores. (Ver figura 1.)

Figura 1: Resultados de la Generación Automática de Entornos a partir del framework SpeedTreeRT y el videojuego la Era de los Imperios II respectivamente.

3.2.1. Espacios Temporales de Generación.

Los generadores se clasifican según el espacio temporal en el que se desenvuelven en:

1. Generadores en Pre-Procesamiento
2. Generadores en Tiempo de Ejecución(*runtime*) [Ada02]

En el primer caso se encuentran las herramientas de generación de entornos virtuales, las cuales permiten la creación y configuración de los niveles o escenarios antes de ser cargados y visualizados por la aplicación en *tiempo real*.

Existen innumerables productos de este tipo, por citar solo algunos se podrían mencionar las herramientas de creación de contenidos de la compañía PRESAGIS [Cor08b], destinadas a la construcción de entornos para simuladores esencialmente. Además se encuentran las herramientas de edición de niveles para videojuegos, entre las cuales resaltan los editores de la compañía Blizzar, usados en sagas como StarCraft, WarCraft y World of WarCraft (WoW) [Cor08a]. De igual forma se pueden enmarcar en esta categoría a los sistemas que en el proceso de carga de los contenidos del escenario generan el entorno según ciertas reglas, como son la extensión del entorno, la cantidad de usuarios, oponentes, vidas y bonificaciones, por lo general son aplicados en videojuegos como DarkForce [Ada02] y Age of Empire [Sta04].

Por otra parte, la generación en *runtime* confiere al espacio de ejecución su lugar de acción, permitiendo la construcción del entorno según sea el nivel de avance sobre el mismo. Es importante puntualizar que todos los generadores en tiempo de ejecución conocidos [GPSG03] [Cor08a] [Sta04] [Dym09] [F.S07], utilizan o proponen algoritmos y técnicas eficientes que permiten una respuesta en *tiempo real*, además de conferirle a los entornos generados un alto nivel de variabilidad por su fuerte componente aleatorio, un elemento esencial en los algoritmos empleados.

Llegar a una conclusión sobre cual espacio de generación es Mejor es complejo, lo cierto es que ambos, según sea la aplicación, pueden utilizarse por separado o en conjunto para lograr el objetivo, por ejemplo: los simuladores que recrean entornos reales, necesitan primeramente realizar la generación en preprocesamiento del terreno y de los contenidos característicos del entorno como árboles, construcciones etc..., los cuales tienen asociados una alta complejidad

computacional, luego en *runtime* se lleva a cabo la construcción del escenario a partir de la distribución espacial de estos contenidos y de otros que pueden ser creados en este momento dado a su baja complejidad operacional, todo a medida que se avanza por el escenario.

Por otra parte valorando la clasificación de Generación en Tiempo de Ejecución, se identifica como una clasificación muy amplia si se conoce que cuando se desea generar entornos en este tiempo se realiza con vista a lograr generaciones que permitan una visualización en *tiempo real*, los generadores con este carácter siempre referencian el logro de este objetivo, por tanto una clasificación más específica puede ser Generación en Tiempo Real. En lo adelante cualquier referencia a este tipo de generación se abordará de tal manera.

3.2.2. Generación en Tiempo Real

La Visualización en Tiempo Real (Real Time Rendering) es descrito por Akenine Moeller y Eric Haines en [MH02] como un proceso en donde el usuario reacciona y actúa ante un coherente cambio de imágenes. El ciclo de interacción entre el usuario y la imagen dibujada es medido en cuadros por segundos (fps), y estos para que sean interpretados en *tiempo real* deben oscilar entre los 15 y 72 fps. Otros autores [nez05] [Boo]colocan la frecuencia de visualización en tiempo real por encima de los 30 fps, lo que hace más natural el proceso de interacción entre el usuario y las imágenes mostradas. El autor de esta memoria ha constatado en la realización de simuladores de conducción que a partir de este valor es que se logra una verdadera senciación de *inmersión*.

La generación de entornos en *tiempo real* implica cumplir con este principio al

ejecutar su proceso, por lo tanto, toda técnica u algoritmo tiene muy en cuenta el factor *eficiencia*, asociado en muchos casos a la reutilización de los contenidos (optimización de los recursos de memoria) y a la complejidad operacional (optimización de los recursos de procesamiento) de estos. Más adelante se explicarán cuales son las técnicas empleadas para la generación de entornos en *tiempo real*.

3.2.3. La Aleatoriedad en la Generación

"La Aleatoriedad en la computadora nunca es una verdadera aleatoriedad" [Haa99]esta afirmación se basa en que para generar números aleatorios por ordenador siempre se debe definir una semilla de generación (random seed), si esta semilla no es variada la secuencia de números generados será siempre la misma, en cambio si se varía constantemente esta se puede aproximar la generación al tipo de aleatoriedad que se evidencia en el mundo real. Partiendo de esto, algunos autores [GS04] dividen los generadores de números aleatorios en dos categorías:

- Generador de números aleatorios verdaderos
- Generador de números aleatorios simulados

El primer caso se refiere a aquellos generadores que producen siempre secuencias diferentes de números aleatorios, estos nunca reproducen una secuencia de números aleatorios generados anteriormente. Por otra parte, un generador de números aleatorios simulado, devuelve siempre una misma secuencia de números aleatorios en cada ejecución dado a que su semilla se mantiene invariable. En la generación de entornos esta última es la más usada, debido a

que se desea mantener por lo general, una coherencia en los entornos tanto en la forma de los contenidos como en su ubicación espacial. Entornos reproducibles constituye el resultado más importante de los generadores de números aleatorios simulados. Por otra parte, la generación de secuencias de números completamente diferentes en cada ejecución no es realista si para ello se toman semillas de generación similares. Con vista a resolver tal problemática se utiliza el Algoritmo Hashing[GPSG03] como posible solución.

3.2.4. Técnicas para la Generación de Contenidos

Los contenidos u objetos que conforman un entorno virtual ocupan un espacio importante en el proceso de generación de entornos virtuales. En va-rios sistemas [Noc05] [Sta04] los contenidos son construidos en pre-procesamiento por diseñadores y herramientas, de manera manual o *procedural* (algorítmicamente), para luego ser cargados en la aplicación en *tiempo real* y colocados según la estrategia de generación seleccionada. En otros [GPSG03] [Dym09] [F.S07] [F.S07] los contenidos son construidos de forma *procedural* en *tiempo real*. Ambas formas de generación se ajustan al tipo de aplicación: las que recrean entornos reales, por ejemplos sistemas geoespaciales 3D [DME07] o sistemas de generación de edificaciones arquitectónicamente reguladas [MWH$^+$06], llevan a cabo la construcción de los contenidos proceduralmente en pre-procesamiento y otras lo procuran *tiempo real* para lograr un nivel de interactividad mayor [LWW08] [F.S07]. Entre las técnicas utilizadas para la generación de contenidos se encuentran:

Ruidos (Noise): por un noise se entiende una función irregular primitiva usada para romper la monotonía de patrones. David S. Ebert con su trabajo

"Texturing and Modeling, a Procedural Approach"(1998) propuso uno de los más populares noise, Perlin Noise, el cual retorna un número real aleatorio entre -1 y 1 dado varios valores de entrada. Estas entradas pueden ser las coordenadas r y t de un particular pixel en una textura o las coordenadas (x,y,z) de un punto en el espacio. Estos han sido aplicados en la generación de texturas y geometrías como: paisajes montañosos, nubes y materiales como la madera y el mármol.

Fractal: Un fractal está definido como una forma similar a sí misma a diferentes escalas. Este se crea por sucesivas réplicas de la misma forma siguiendo diferentes reglas. Probablemente el ejemplo más simple de forma fractal sea la curva de Koch, en la cual una forma geométrica es subdividida en segmentos en los cuales cada segmento es remplazado con la forma inicial y así sucesivamente. Un Fractal puede ser generado a partir de procesos iterativos o recursivos. Se utilizan en la generación de árboles, y terrenos [Pub03].

Mapa de Alturas: Es una técnica de generación muy utiliza esencialmente en la creación de terrenos. El mapa es un arreglo bidimensional de valores de altura organizados en una rejilla regular. En cada par (x,y) de la rejilla se almacena el valor de z (altura). Este valor coincide con la coloración en escala de grises del pixel (x,y) de la imagen que se tome como mapa, el cual se acota en el intervalo entre 0 que sería la altura más baja(color negro) del terreno, y 255, la más alta(color blanco). Esta imagen puede ser creada en cualquier editor que permita la creación de imágenes en escala de grises. El método es genérico y permite generar un número interminable de terrenos, solo es necesario proporcionar la imagen en escala de grises y las cotas de altura asociadas a los valores 0 y 255 respectivamente [Pub03].

Gramáticas: Esta técnica hace uso de principios para la especificación de lenguajes, en donde sus componentes pueden ser cualquier tipo de elementos, dígase letras, palabras o geometrías y texturas cuando se habla de generación de contenidos. Las reglas de producción que la conforman establecen las expansiones de los no terminales sucesivamente hasta culminar en la aplicación de una regla cuyo resultado es un terminal. Gramáticas de Geometrías y de Texturas son utilizadas [MWH+06] [LWW08] [F.S07]para la generación de edificios de diferentes tipos de arquitecturas, así como modelos de fachadas ajustadas a principios arquitectónicos. También pueden utilizarse para la creación de árboles. Su principal complejidad radica en definir una buena gramática, lo suficientemente genérica para generar una gran variedad de contenidos de un tipo determinado.

Sistema de Losas (System Tiling): Esta técnica de generación se basa en pequeños polígonos regulares tales como triángulos, cuadrados, hexágonos etc..., llamados losas, los cuales son conectados en forma matricial para conformar una gran estructura. Es utilizado esencialmente en juegos 2D en donde se necesita crear grandes terrenos a partir de un conjunto de pequeñas imágenes, las cuales son reutilizadas para recrear extensiones de diferentes características. En juegos como la Era de los Imperios y Diablo se pueden observar sus resultados. Este método ha tenido extensiones a los espacios 3D como se puede ver en [Noc05] [GPSG03] con buenos resultados, su principal inconveniente es lograr la homogeneidad del terreno, lo cual es afectado en ocasiones por las divisiones entre losa y losa. Trabajos sobre el mapeo de texturas y remodelación de geometrías pueden resolver tal problema.

3.2.5. Estrategias de Generación de Espacios Virtuales

Las estrategias de generación de entornos virtuales se basan esencialmente en la forma en que los contenidos serán dispuestos en el entorno. Por tal motivo, estas pueden incluir o no la generación procedural de contenidos. Otro elemento esencial en estas estrategias, es la importancia que se le presta a la reutilización de los contenidos en sí, o de las partes (geometrías bases y texturas) que conforman a estos, todo con el fin de lograr un mejor nivel de eficiencia [GS04].

Estrategia *Llenado del Volumen de Visión*

Llenado del Volumen de Visión(Frustum Filling) es una de las estrategias más acertadas en cuanto a eficiencia, su principio es llenar constantemente el *Frustum* con contenidos, los cuales al salir del mismo son elminados producto del desplazamiento y/o cambios de orientación de la cámara. Esta estrategia propuesta por S.Greuter y N.Steward en [GPSG03] (Ver figura 2) es una de la candidatas más importantes cuando se habla de generación en *tiempo real*, esencialmente por su economía de recursos. Su principal inconveniente es que su integración con sistemas como motores gráficos resulta complejo ya que introduce un nuevo elemento en el proceso de visualización; antes de realizar el Frustum Culling es necesario realizar el Frustum Filling, esto conlleva a que para integrarlo a un *motor gráfico* sea necesario modificar el proceso de visualización del mismo, algo que no es simple. Los sistemas encontrados que hacen uso de esta [GPSG03] [Noc05], contienen varios componentes comprendidos en los motores gráficos que hacen de estos difíciles de reutilizar si se persigue un enfoque de integración genérico, multi-motor gráfico. Los entornos que se han generardo haciendo uso de esta son: ciudades, paisajes y entornos espaciales,

todos pueden ser generados de forma infinita.

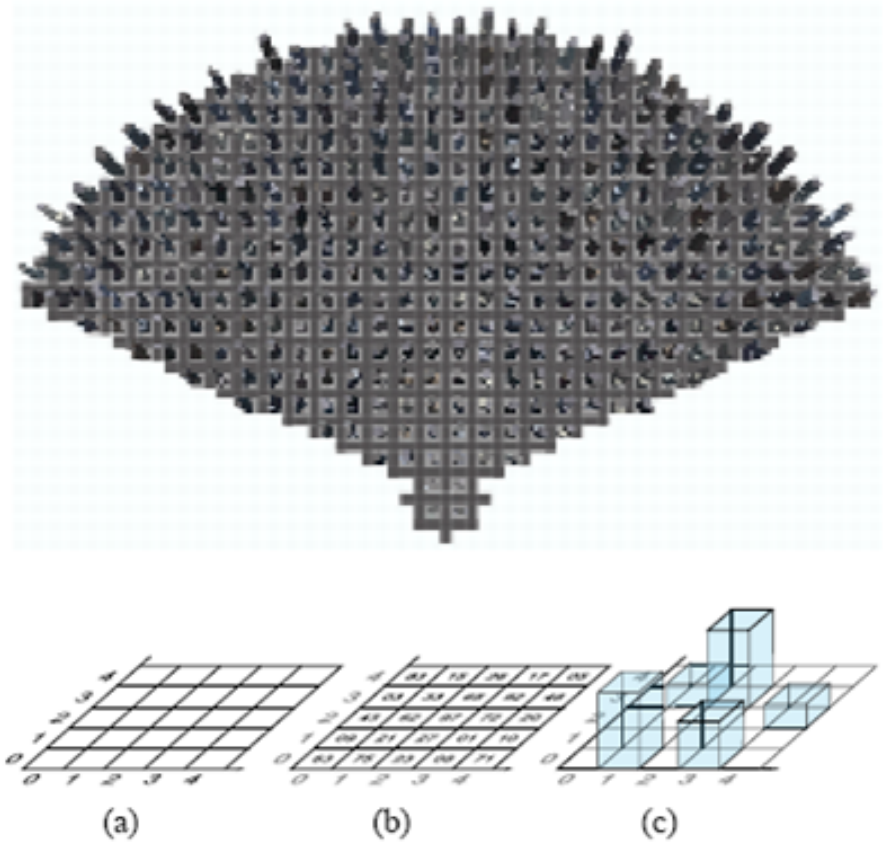

Figura 2: La figura muestra primeramente el resultado del Proceso del *Frustum Filling*. Para ello se basa en una rejilla regular(a) cuyas celdas son encuestadas para saber si se encuentran dentro del *Frustum*, las que cumplen el requisito son identificadas como espacio para colocar contenidos, en cuyo caso son generados de forma procedural(c) usando la semilla almacenada en estas.

Estrategia Basada en Relatividad del Movimiento

La esencia de esta estrategia no se ha encontrado reflejada en ninguna publicación de las estudiadas en este trabajo, pero se aplica en videojuegos sobre todo de corte espacial como el Bang [Int]. En esta clase de videojuegos el movimiento de los contenidos es un elemento muy marcado y las geometrías por lo general son a bajo polígono, algo que ayuda a una mejor aplicación y a resultados en *tiempo real.* Los contenidos son posicionados constantemente, en

su lugar la cámara se mantiene estática, solo se reorienta; partiendo de esto el movimiento de los objetos respecto a ella está dado por:

$$\overrightarrow{velocidad} = \overrightarrow{velocidadcamara} + \overrightarrow{velocidadcontenido}$$

Como se puede apreciar solo sigue un principio básico de la mecánica clásica. Una de sus ventajas es la flexibilidad para la reutilización, además de que permite recrear entornos infinitos.

Estrategia Gestor de Página(Page Manager)

Esta estrategia utilizada por M.Danaher para la creación de terrenos 3D geo-referenciados [Mau03] constituye una técnica diferente a las utilizadas hasta el entonces para esta clase de aplicación. Esta subdivide el mapa a generar en pequeños submapas, los cuales están conformados por pequeños bloques llamados páginas. Como se puede observar en la figura 3,las páginas, se ordenan de forma matricial y conforman el terreno, lo cual permite identificar de una manera rápida en que posición se encuentra la cámara y marcar así las páginas más próximas a esta para su posteriro generación, las restantes que no cumplen esta condición son eliminadas. A diferenicia de las otras técnicas, esta abordan la generación basada en páginas de otra forma; no se centra solamente en los contenidos de la página en donde se encuentra la cámara, sino también en los contenidos pertenecientes a las páginas vecinas.

Otro punto importante es el uso del método de envoltura esférica, creado por el autor, el cual permite no romper el efecto de continuidad en un mapa limitado, y reutilizar una misma estructura del gestor de página para generar coherentemente to do el entorno. Como se puede apreciar en la figura al pasar

del bloque 5 al 6 ocurre que se generan las páginas pertenecientes a los bloques 1, 4, 7; al otro extremo sí, pero en el submapa siguiente. Al realizar este trabajo internamete, la continuidad es garantizada. La técnia es respaldada por la optimización de la visualización basada en niveles de detalles, lo cual se realiza solamente en el submapa en donde se ecuentra la cámara y garatiza resultados en *tiempo real* al decir del autor, pero no se visualiza gráficas de análisis de rendimiento.

Figura 3: Progreso de la estrategia en dos avances de página a la derecha. El color verde representa la página en donde está la cámara y las amarillas las páginas activas, hacia donde se puede desplazar el usuario y las generadas según el submapa.

A los efectos de este trabajo la técnica puede aplicarse en videojuegos. Además, brinda posibilidades para generar entornos infinitos ya que el método de envoltura esférica brinda esta posibilidad; solo existe el problema que en un tiempo determinado el terreno volverá a repetirse. Por otra parte, el criterio que se utilizado para determinar las páginas más próximas limita su alcance, este criterio puede traer consigo problemas en cuanto a : números de mayas a generar y almacenar en memoria, ya que se basa en bloques muy grandes de contenidos, en donde el volumen de visión no sobrepasará el próximo nivel de páginas según la posición y dirección de la cámara. Con páginas de menor tamaño esto puede mejorar pero el sistema romperá la sensación de continuidad dado esto.

Estrategia Basada en Gramáticas

En otro orden y complejidad de generación se encuentra la presente estrategia, la cual usa como base de generación gramáticas, las cuales como se mencionó anteriormente pueden ser utilizadas para generar contenidos y también para definir las diferentes características topológicas de un tipo de entorno, sustentándose esencialmente en la disposición de los contenidos según ciertas reglas. Las gramáticas libres de contexto se recomiendan [Ada02] como potenciales candidatas para esta estrategia. Con el objetivo de comprender con mayor rigor estos elementos se presentará el siguiente ejemplo que constituye una versión del mostrado en [Ada02].

Se desea generar un entorno de interiores para un videojuego (Ver figura 4) en donde los contenidos de la lógica del videojuego (oponentes, vidas, bonos) son colocados de manera aleatoria. Para llevar a cabo este proceso se necesita primeramente: generar el entorno de interiores y luego, colocar los contenidos. El primer paso puede ser resultado de usar una gramática de grafos [Ada02] en la cual se establece las interrelaciones posibles entres las diferentes habitaciones, en este caso llamadas Secciones (Ver figura. 5a). Para el segundo, se crean gramáticas de contenidos cuyo principio sería construir aleatoriamente cierta cantidad de estos elementos. Vale la pena analizar que el ejemplo no incluye las interrelaciones que pueden existir entre los contenidos de la lógica de un juego lo cual puede dar como resultado gramáticas más complejas que permite generar niveles más variables.(Ver figura. 5b)

Los resultados más significativos han sido en la Generación en Pre-Procesamiento, en la cual es muy aplicada [Ada02] [DME07] [Pub03], pero ya existen trabajos que la incorporan en *tiempo real* para la generación de entornos de interiores [HBW06].

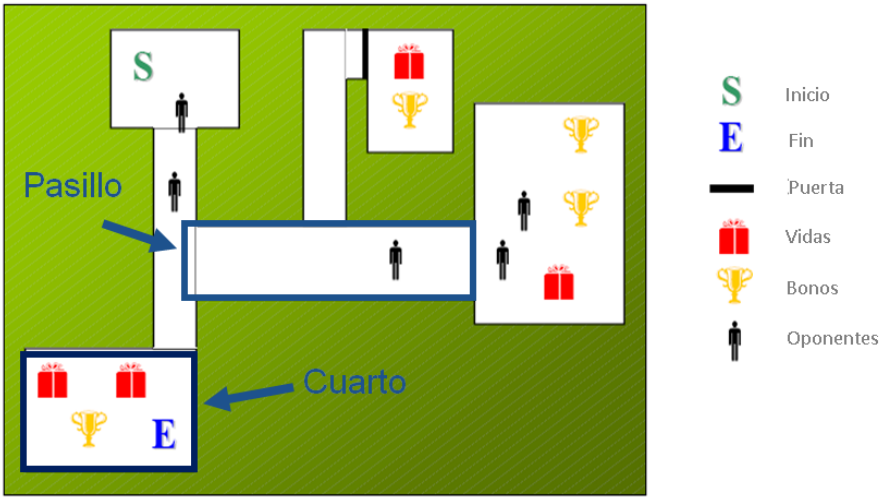

Figura 4: Prototipo de Entorno de Interiores para un VideoJuego.

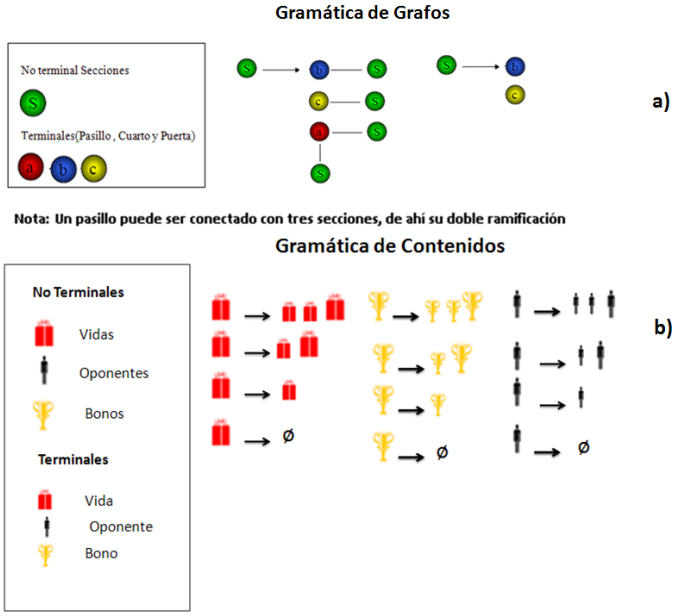

Figura 5: *Gramáticas libres de Contexto*. a) Gramática de Grafos para generar el entorno de interior y b) Gramática de Contenidos para generar los elementos de la lógica de un videojuego.

3.2.6. Sistemas de Generación en Tiempo Real. Funcionalidades y Arquitectura.

Los sistemas de generación en *tiempo real* son muy frecuentes sobre todo cuando se trabaja en videojuegos. Su esencia, por lo general, es permitir la construcción de entornos complejos. En este sentido se puede encontrar a SpeedTreeRT [Vis], un sistema de generación construido por Interactive Date Visualization que da soporte a la generación procedural de contenidos y de entornos. Ha sido integrado dentro del Epics Unreal Engine y en otros *motores gráficos* para videojuegos con buenos resultados. Permite una frecuencia de visualización equilibrada para lo cual está equipado de buenas técnicas y algoritmos de optimización.

Por otra parte se puede encontrar a DescensorEngine [Worc] construido por Binary Worlds, el cual provee una solución integradora para la construcción procedural de entornos 3D en *tiempo real*. Permite recrear paisajes con contenidos como casas, calles, edificios y árboles. El proceso de generación es controlado por parámetros, los cuales pueden modificarse por el usuario a partir de un editor. Todas las geometrías son generadas y organizadas en una estructura de árbol y solo se genera contenidos visibles a la cámara. Logra una frecuencia de visualización estable, para lo cual dibuja las geometrías en variados niveles de detalles, haciendo uso de impostores, y apoyándose en diferentes técnicas de selección de visibilidad. El producto ha sido utilizado en la elaboración de videojuegos igualmente. Ambos generadores están dentro de los generadores comerciales, sus arquitecturas no son provistas para un análisis.

En el plano no comercial se encuentran otros sistemas de generación en *tiempo*

real, hasta lo que se conoce ninguno provee el código pero se expone más su buen hacer mostrando los algoritmos, técnicas y arquitecturas en que se basan. En este sentido se encuentra el generador propuesto por Greuter y Stewart [GPSG03] [SJ03], el cual está preparado para la creación de cualquier tipo de entorno al decir de los autores, aunque las pruebas se basan en la creación de ciudades infinitas solamente. Como la mayoría de lo generadores estudiados genera contenidos hacia la Cámara, en este caso basándose en la estrategia de generación *Frustum Filling* y construyendo edificios, calles y árboles de manera procedural, con algoritmos que proveen un nivel de variabilidad muy alto como el Algoritmo Hashing. Utilizan a OpenGl como API de visualización.

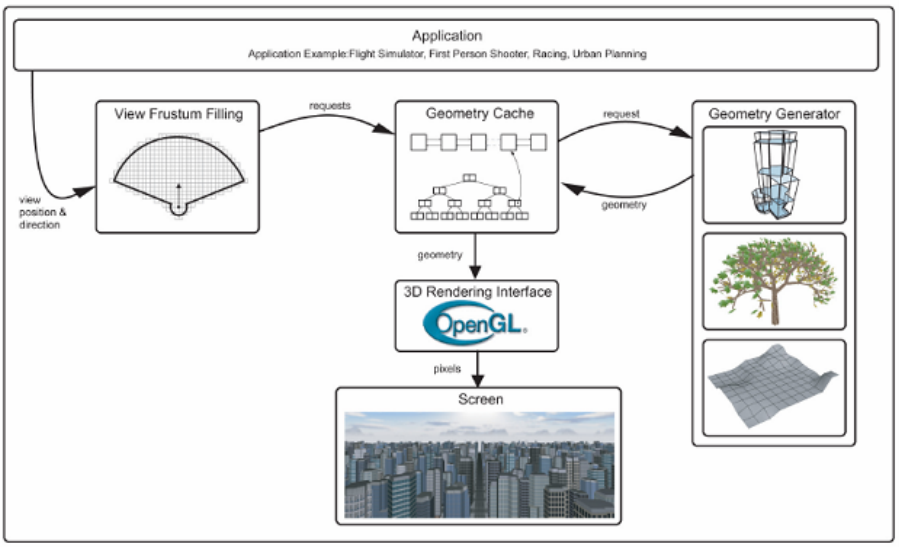

Figura 6: Arquitectura del Framework para la generación de entornos en tiempo real.

Continuando el análisis en este sentido se puede encontrar el generador Scaper [Noc05] el cual basa su sistema en la estrategia de *Frustum Filling* también, algo que lo limita en cuanto a flexibilidad. En este caso no utiliza la generación procedural de contenidos, solo se centra en la disposición de los mismos en el entorno haciendo variaciones en forma(escalado), posición y orientación, para

lo cual cuenta con un robusto fichero XML que da soporte a toda una variedad de contenidos y alternativas de generación de entornos virtuales. Es un sistema que genera los entornos en función de la cámara basándose en estructuras como los Generadores, elementos principales que controlan una sección espacial según su Topología[3], en la misma distribuyen los contenidos. Para lograr una visualización en *tiempo real* se apoya en técnicas de optimización como niveles de detalles y *Frustum Culling*. Sus potencialidades se han probado en entornos de ciudad, paisajes y espaciales. Presenta una arquitectura flexible y extensible (Ver figura 7)para la confección de otros entornos.

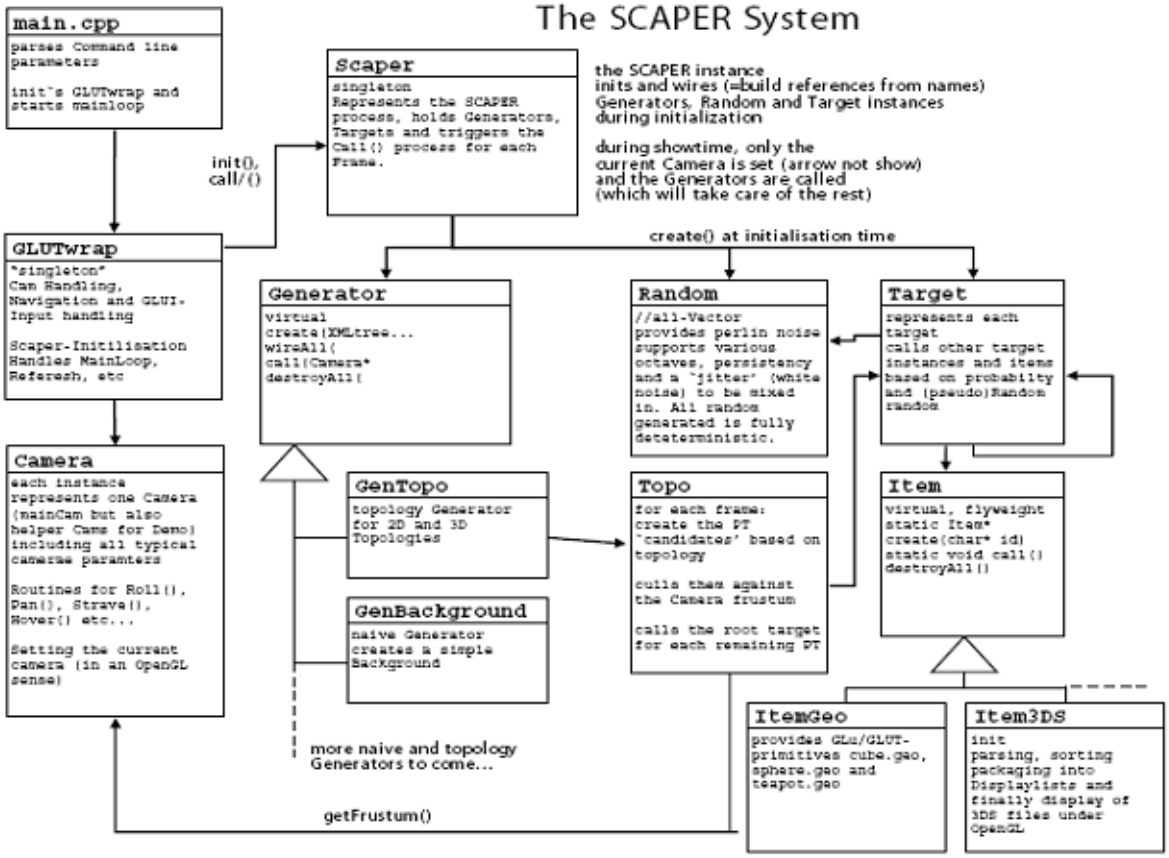

Figura 7: Arquitectura del Sistema Scaper.

[3]La Topología identifica el tipo de volúmen de encierro del espacio de generación(rectángulo, esfera,cubo etc...)

3.3. Solución Técnica a la Generación de Entornos en la Herramienta SceneTollkit.

La investigación realizada para definir el sistema fue amplia. En esta se identificó que existen muchos sistemas de generación en *tiempo real* elaborados, pero publicados en profundidad con sus principios y técnicas empleadas solo dos: el Framework para la Generación de Entronos Virtuales y el Sistema SCAPER.

Todos los generadores, dígase comerciales o no, presentan una tendencia a la especialización en la generación de ciertos tipos de entornos. Esto presupone de alguna manera que los principios y estructuras de los generadores no son lo suficientemente genéricas para permitir la creación de cualquier entorno basado en diferentes estrategias de generación. Además desde el punto de vista ingenieril, los generadores no comerciales antes mencionados, no presentan una arquitectura libre para el acople con diferentes *motores gráficos*, un aspecto muy importante para el presente trabajo. Ante esto se decidió elaborar un sistema capaz de incorporar lo mejor de lo estudiado y otros conceptos para un carácter más genérico.

3.3.1. Elementos Esenciales de la Propuesta

El sistema agrupa un grupo de conceptos sin cuya definición sería compleja la comprensión de su funcionamiento. Estos son:

Contenido: Por contenido se entiende todas aquellas geometrías que conforman un entorno virtual

Estrategia de Generación: Se le denomina a las técnicas de generación de contenidos y de entornos virtuales.

Capa de Generación: Elemento encargado de efectuar el proceso de generación tanto de contenidos como de una porción de entorno basado en alguna estrategia definida y tomando en consideración el espacio que controla según su Topología.

Topología de una Capa: Encierra la forma que esta tiene y delimita el espacio que controla, puede ser 2D o 3D. Las topologías pueden ser capas 2D con forma de cualquier primitiva (Puto, Rectángulo, Circunferencia, Triángulo...). De igual forma las 3D (Cubo, Esfera...).

Generador de Entornos: Elemento encargado de orquestar todas las capas dentro de un entorno con vista a lograr la generación del mismo.

Cámara: Representa al usuario en el sistema y es un punto esencial para la generación en tiempo real.

Memoria a Corto Plazo: Este componente constituye un elemento propio de la inteligencia artificial expuesto con mayor claridad en [Rav04] el cual es utilizado por los *agentes inteligentes* para la toma de decisiones. En este contexto le provee al generador un espacio de almacenamiento de información que se borra cada cierto período de tiempo y recoge aquellos contenidos que son significativos para la aplicación, ejemplo : los elementos de la lógica de un videojuego. Este elemento permite establecer una *coherencia estructural* en los entornos generados.

Representación gráfica del generador:

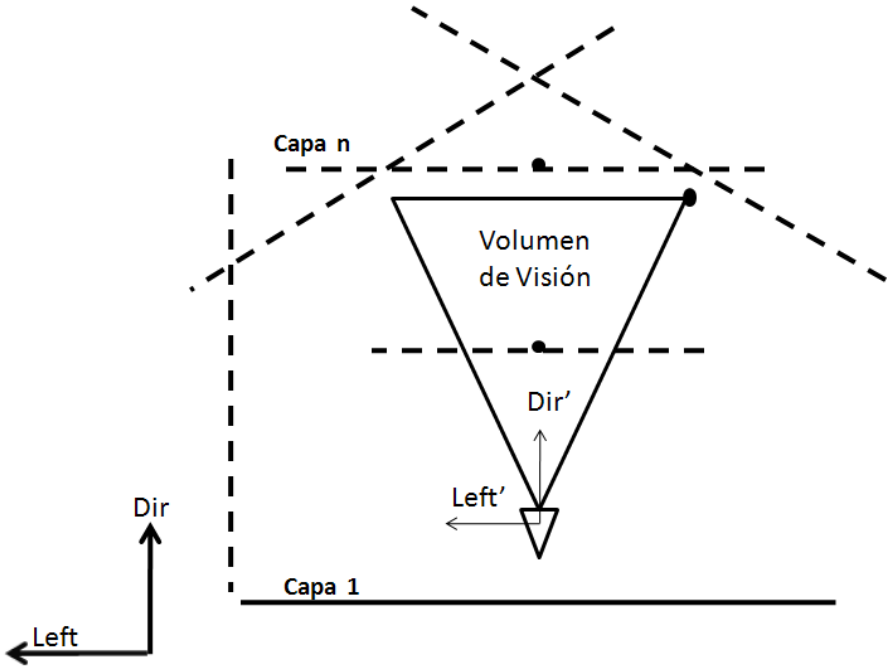

Figura 8: Esquema del generador en donde se visualiza la libertad de orientación de las capas y el principio de orientación según los tres vectores directores (Dir,Left,Up). Este elemento permite un nivel de generalidad respecto al convenio de orientación que se seleccione.

Como se puede observar las capas pueden orientarse respecto a cualquier sistema de referencia e incluso trasladarse espacialmente. El sistema de generación siempre se traslada relativamente respecto a la Cámara lo que permite lograr una estabilidad en el número de capas a controlar, y generar contenidos en posiciones ventajosas para su visibilidad. Todas las capas son controladas por el Generador de Entornos según la estrategia de generación establecida. Es importante señalar que el sistema permite el uso de la aleatoriedad simulada y verdadera según se desee.

El ciclo de generación se ejecuta en todos los frames. El orden del proceso es:

1. Se actualizan todas las Capas relativamente a la Cámara según la Estrategia de Generación definida.

2. Se realiza un chequeo de colisiones entre capas. En los casos de colisión se corrigen si la estrategia no la concibe como parte de ella.

3. Se ordena a cada capa generar los contenidos y posicionarlos o posicionarlos solamente según su topología y estrategia definida. En este punto el generador selecciona si en el espacio de generación se debe posicionar nuevos contenidos o colocar algunos almacenados en la Memoria a Corto Plazo.

4. Se actualiza el *tiempo de vida* de la memoria. En caso de que este haya llegado a su fin se limpia la memoria y se condiciona para almacenar nuevos elementos.

En otro orden de importancia el sistema brinda un conjunto de elementos conceptuales como *Contenido* y *Cámara* que posibilitan abstraer el proceso de generación del proceso de visualización, lo cual permitió una integración rápida con la STK y la posibilidad de su integración con otros motores. De igual forma el módulo contiene un sistema de memoria, llamado Memoria a Corto Plazo que parte de una de las carácterísticas que pueden tener los *Agentes Inteligentes*, posibilitando que el generador tome en consideración la permanencia de un contenido asociado o no a la lógica del videojuego en el entorno, al disponerlo en el mismo lugar tal y como fue definido en su primer avistamiento. Esto permite establecer una *coherencia estructural* en los entornos generados.

3.3.2. Diseño del Sistema

Para el diseño del módulo (Ver figura 9) se siguió el paradigma orientado a objeto y se aplicó diferentes patrones como el SingleFactory, Facade, Expert, entre otros, todos regulados a partir de los principios arquitectónicos de la Herramienta. Los componentes arquitectónicamente más significativos y sus responsabilidades serán expuestos a continuación:

CCamera y CContent: Ambos son tipos propios del generador, identifican a los elementos Cámara y Contenido respectivamente. Estos sirven de base para crear interfaces de conexión con los motores gráficos. En este caso solo se crean las interfaces para la STK, pero este principio ingenieril hace del módulo extensible y reutilizable.

CFactory : Elemento principal que representa el patrón SingleFactory, de él se puede generar diferentes factorías especializadas, esencialmente cuando se intente conectar a otro *motor gráfico*. Solo por el momento se crea la factoría CSTKFactory.

CStrategy: Permite la creación de componentes que encapsulan diferentes algoritmos de generación. Actualmente solo soporta estrategias que sirven de soporte a las estrategias de generación de entornos, las cuales pueden ser utilizadas en cualquier generador específico (CSpaceGenerator o CLanSpaceGenerator).

CLayer: Este componente encapsula el elemento Capa y como tal tiene la funcionalidad de generar contenidos de forma procedural y realizar transformaciones espaciales (traslación,rotación,escalado) sobre los mismos. Todo depende de las estrategias que la componen.

CShortMemory: Este elemento permite crear diferentes tipos de memorias a corto plazo como la CGridMemory basada en la estructura de datos espacial Rejilla Regular.

CEnviromenteGenerator: Este componente permite la creación de generadores tipos que pueden basarse en cualquier estrategia de generación de entornos.

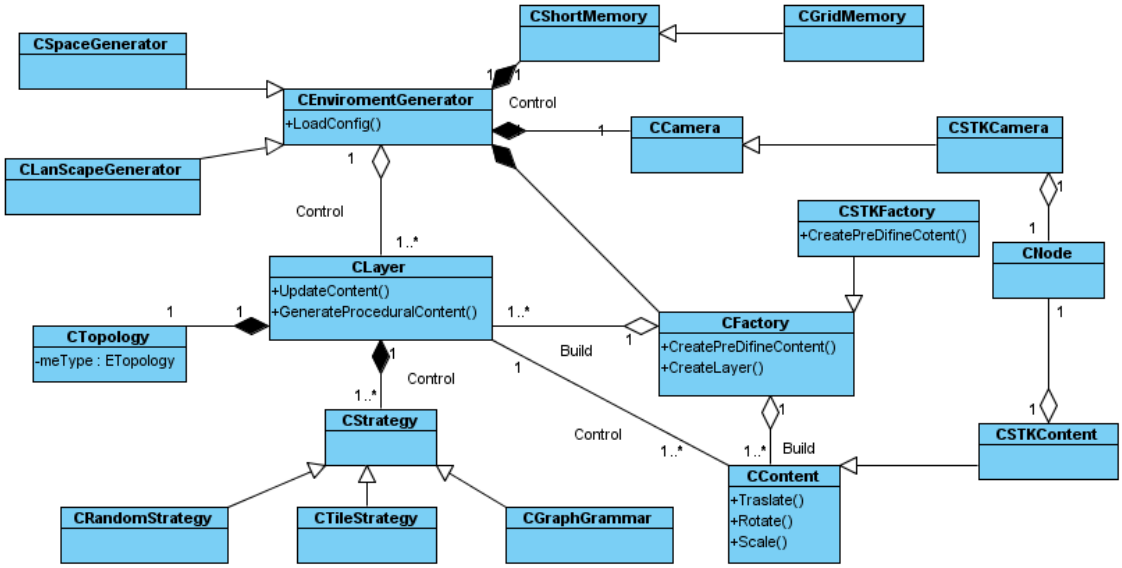

Figura 9: Diseño del Módulo de Generación de Entornos

En otro orden de diseño se desarrolló un sistema de fichero para darle soporte a dicho generador (Ver figura 10), el mismo solo permite por el momento configurar las capas, dígase establecer respecto a que sistema de referencia se va orientar, posición espacial inicial, contenidos predefinidos a generar etc... En posteriores trabajos se perfeccionará con vista a que presente un carácter de script avanzado, similar en flexibilidad al propuesto por el sistema Scaper.

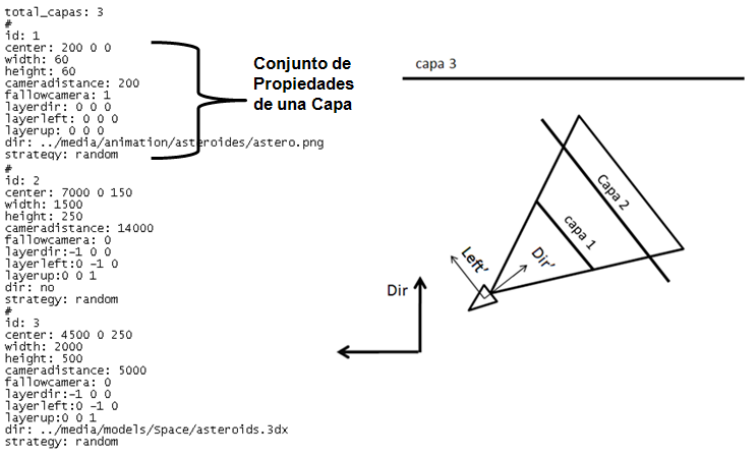

Figura 10: Fichero de configuración acorde al generador espacial

3.3.3. Solución para la generación de entornos espaciales y de paisajes

Como se observó anteriormente el sistema constituye una base genérica para definir tantos generadores de entornos como se requieran. Por lo general se ajustan a un tipo de entorno y a una estrategia de generación específica, aunque se pueden mezclar si se desea, la estructura lo permite. A continuación se procede a la explicación de dos generadores específico: SpaceGenerator y LanScapeGenerator.

El primero basa su funcionamiento en la Estrategia Basada en Relatividad del Movimiento la cual le permite generar entornos espaciales infinitos y da soporte a la reutilización de contenidos. Este está compuesto por tres capas dispuestas como se observa en la figura 11. Las mismas están establecidas de acuerdo a las necesidades del videojuego Meteorix, pero puede ser reutilizado e incluso aumentar sus capas para generar más contenidos como planetas, agujeros negros etc..., solo es cuestión de configurarlo para los nuevos elementos

de generación. En este sentido se puede usar el fichero de configuración, el cual brinda flexibilidad para ello.

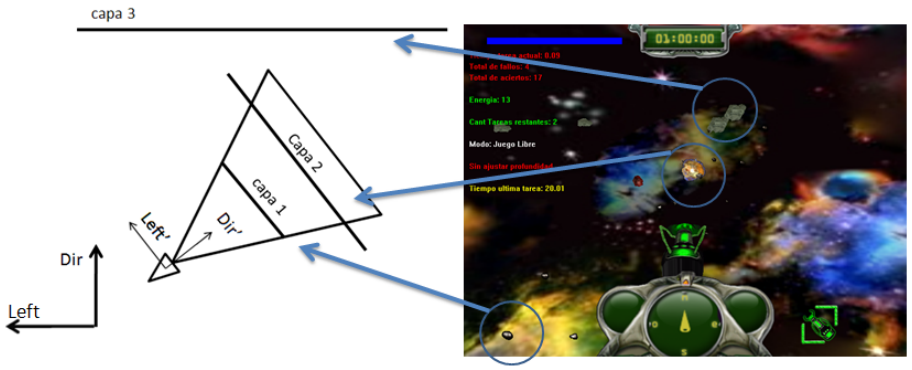

Figura 11: La capa 1 y 3 generan contenidos de ambientación (asteroides grandes y pequeños), en el caso de la Capa 2 genera los contenidos relacionados con la lógica del Juego (Asteroides de Color).

Por otra parte el generador de paisajes encapsula una estrategia de generación llamada Losas Dinámicas(Dynamic Tiles) con puntos en común con las Estrategias Sistemas de Losas y Gestor de Páginas. En esta estrategia, al igual que con todas las que realizan la generación en *tiempo real*, toma en consideración el desplazamiento de la Cámara(usuario) por el entorno, y en la medida que esto se realiza ajusta el sistema de losas a su alrededor. En el proceso las losas son trasladadas y reorientadas, según sea la dirección del movimiento. Con esto se garantiza un ahorro de los contenidos de diseño y un sistema estable en cuanto a su configuración. En la figura 12 se muestra el resultado de esta propuesta a partir de dos iteraciones del sistema de generación. En este caso, el sistema realiza una rotación por columnas. Como se puede apreciar el sistema se encuentra siempre posicionado en función de la Cámara. La Cámara permanece siempre dentro de la submatriz 3x3, cuyos componentes en las filas y columnas 0 y 2 representan un volumen de encierro. La proximidad a las fronteras de dicho volumen determinan cuando se genera y también el sentido

de la generación.

La dimensión de la matriz debe ser regular *(nxm donde n=m)* y de una dimensión comprendida en el intervalo *(5x5, nxn)*, en este sentido siempre se recomienda dimensiones impares con vista a lograr la matriz de encierro más eficiente *(3x3)*. La distancia de generación *DG*, valor mediante el cual se obtiene la matriz de encierro a partir de la matriz global*(n- DG x n-DG = 3 x 3)* es: *DG = parteentera(n/3)*.

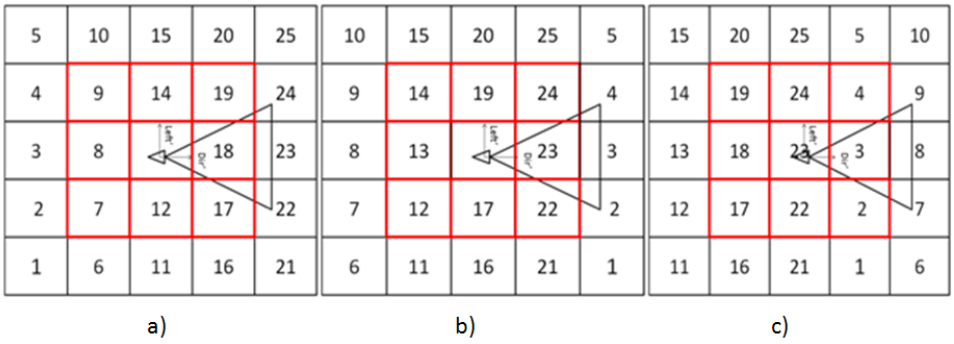

Figura 12: Resultado de la aplicación de la estrategia de generación luego de dos iteraciones. Las losas de color rojo son hacia las cuales se puede desplazar el observador, estas delimitan el volumen de encierro.

La configuración del sistema de generación acorde a la estrategia antes mencionada se puede observar en la figura 13a. Las capas controlan espacios cuadrados dado a su topología, estos son cubiertos por bloques de contenidos conformados por una base de terreno y un conjunto de contenidos que se coloca sobre el mismo, ambos son predefinidos, no se construyen de forma procedural. La propuesta selecciona aleatoriamente la base y el conjunto de contenidos a colocar en la misma, mientras más variantes predefinidas, mayor será el nivel de variabilidad de los entornos generados. La figura 13b muestra un paisaje generado utilizando como base 4 bases de terreno y 3 conjuntos de contenidos a colocar sobre cada una de estas.

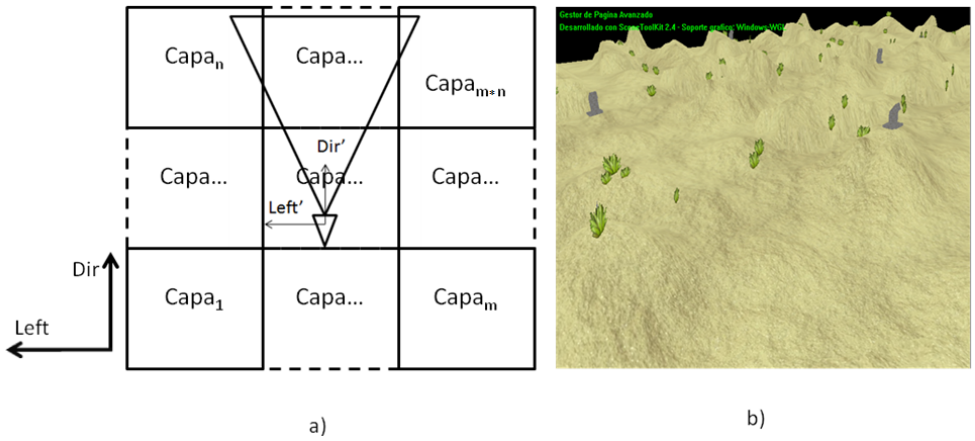

Figura 13: En la figura (a) se presenta la configuración del sistema por capas para aplicar la estrategia Losas Dinámicas. En la (b), el resultado de su aplicación con un sistema de dimensión *7x7* y $DG = 2$.

3.4. Pruebas al Sistema. Mediciones e Interpretaciones

Las pruebas al sistema estuvo en función de comprobar la eficiencia de los dos generadores creados: SpaceGenerator y LanScapeGenerator. Las mismas fueron realizadas sobre una PC ASUS de 1 Gbyte de RAM, procesador Celeron a 1.6 GHz, sin tarjeta de video y con el sistema operativo Window XP SERVIPACK2. La resolución que se estableció para las mismas fue de 800x600.

El SpaceGenerator fue sometido a prueba a partir de la incorporación del mismo en el videojuego Meteorix, en donde se necesitaba recrear entornos espaciales infinitos. Integrado conjuntamente con el proceso de la lógica, se observó que los cuadros por segundos(fps) estaban por encima de 70 todo el tiempo; sin duda el generador responde en *tiempo real*, el cual está dado en parte al alto grado de reutilización que desarrolla y al número de geometrías a visualizar, resta pues continuar probando con entornos más complejos para conocer hasta que punto responde en *tiempo real*.

En el caso del LanScapeGenerator se procedió a generar tres tipos de entornos diferentes, estableciendo para cada uno una semilla de generación constante (aleatoriedad simulada). En todos los entornos se estableció sistemas de losas de diferentes dimensiones y con la Cámara siguiendo la misma trayectoria. También se trabajó con 4 grupos de contenidos conformados cada uno por una base de relieve de forma cuadrada de lado igual 1238 unidades aplicación, y 4 grupos de contenidos a colocar sobre la misma. La lectura de los fps y la densidad poligonal fue recogida cada un segundo, realizando un monto de 20 lecturas, a partir de las cuales se determinó el promedio para cada una de las variables seleccionadas. El primer resultado se evidencia en la siguiente gráfica.

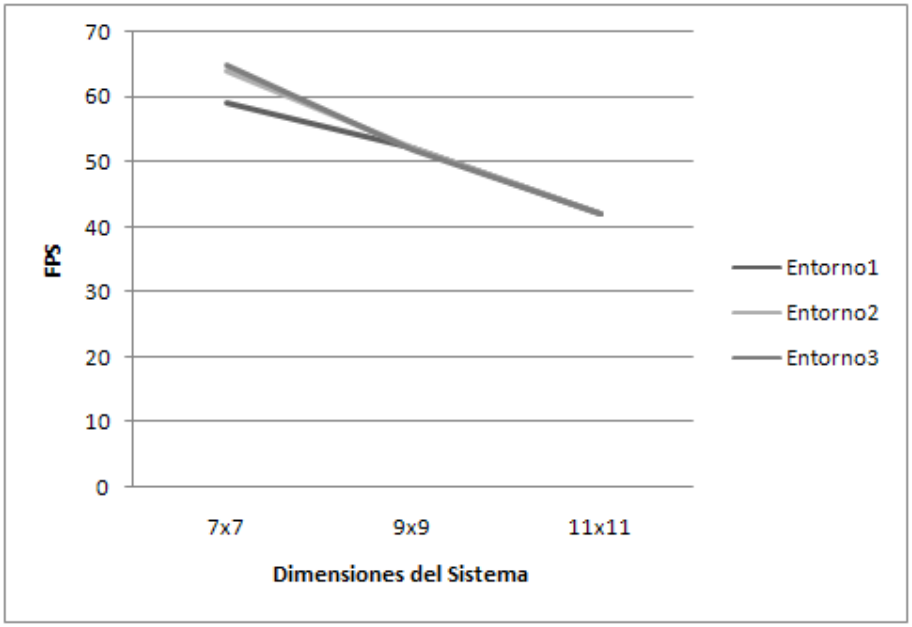

Figura 14: Gráfica de *fps* en función de las dimensiones del Sistema de Losas.

Como se puede apreciar, a pesar de la variabilidad de contenidos lograda, lo cual se constató a partir de observaciones, el sistema respondió en cada entorno prácticamente de la misma forma para cada dimensión abordada. El sentido de similitud de los fps fue el resultado de la estabilidad de la estrategia independientemente del entorno y del *rendering* de densidades poligonales

aproximadamente iguales. Todo esto demuestra que no importa si se varía la semilla de generación, los fps serán aproximadamente iguales y en *tiempo real*, si se basa en iguales conjuntos de contenidos predefinidos con los cuales generar el entorno. Por otra parte, las dimensiónes máximas encuestadas fue de 11x11, esto se debió a que con estas dimensiónes prácticamente era imperceptible el momento en que se generaba el entorno, pero para tener las dimensiónes límite de respuesta en *tiempo real* (para el presente trabajo por encima de 30 fps) basta con analizar que la tendencia de los fps en la gráfica es a disminuir de 10 en 10, de aquí que para dimensiónes de 13x13 la respuesta continua en *tiempo real*(32 fps), no así para 15x15.

Una caída de 10 fps por cada aumento de las dimensiónes constituye otro valor importante a la hora de valorar la eficiencia de la estrategia. En este sentido, se realizó otra prueba sobre el Entorno 1, bajo las mismas condiciones pero ejecutando el proceso de generación sin ejecutar el proceso de rendering, con el objetivo de determinar el % de incidencia del algoritmo implementado en la caída de los fps. Los resultados se observan en la siguiente gráfica.

Al observar estos resultados se nota que sin duda el algoritmo de generación contribuye a esta caída de los fps en la medida que las dimensiónes del sistema aumenta, pero solo en un 30 % cuando se analiza el peor caso (Dimensiónes 11x11). Esto identifica que se debe trabajar para optimizar aún más el algoritmo, pero más que eso, se necesita trabajar en la optimización del proceso de *rendering*, ya sea utilizando técnicas implementadas en la herramienta o implementando otras. Esto resultará de vital importancia para que el generador obtenga resultados en *tiempo real* en entornos más complejos.

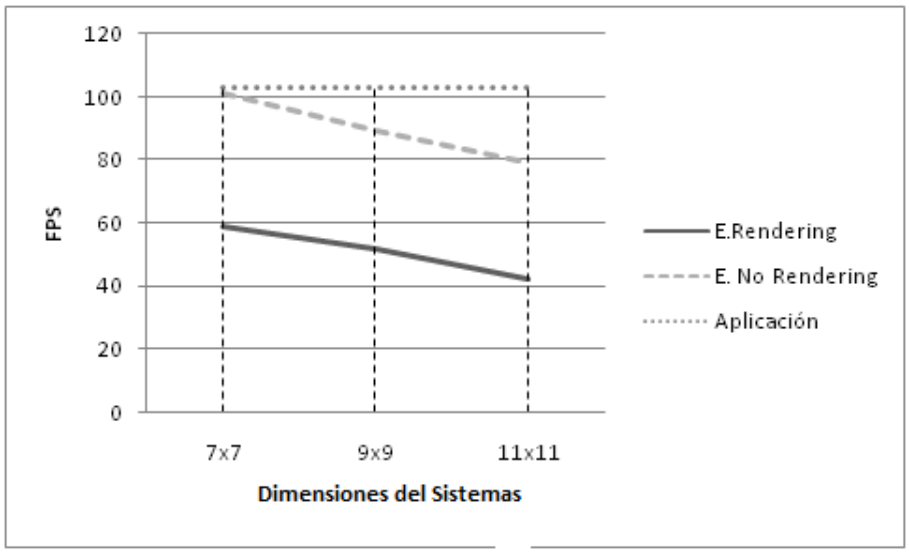

Figura 15: Incidencia de la Estrategia Losas Dinámicas en el proceso de generación con y sin *rendering*.

Aunque el contexto de este trabajo se encuentra enmarcado en el desarrollo de un sistema de generación para la Herramienta STK, otro punto que debe caracterizar a este sistema es su flexibilidad para su integración con diferentes *motores gráficos*. En este contexto se realizó una prueba que llevó a validar este requerimiento. El *motor gráfico* seleccionado fue Ogre3D. Sobre el mismo se desarrolló un ejemplo en donde se integró el generador LandScapeGenerator, el cual mostró resultados satisfactorios en cuanto a flexibilidad, pero no en las respuestas en *tiempo real*. Para las mismas condiciones de hardware anterior el sistema no devolvió resultados satisfactorios; 20 fps fue el mejor resultado en un recorrido libre de la Cámara por el entorno. Las dimensiónes del sistema fue de 7x7, con una base de contenidos conformados por: un terreno y un grupo de contenidos asociados al mismo. A pesar de que estos resultados no fueron realizados con toda la profundidad necesaria, se considera que se llevará a respuestas en *tiempo real* en el mismo instante que se adicione algún componente de optimización que Ogre provee, además de optimizar el diseño.

En la siguiente figura se muestra el ejemplo y se ve la efectividad de la neblina como recurso para enmascarar los límites del sistema y por tanto el momento en que se realiza la generación.

Figura 16: Aplicación del Generador en el Motor Gráfico Ogre3D

4. Conclusiones y recomendaciones

Tomando en consideración el objetivo del presenta trabajo, y los resultados obtenidos, se pude arribar a la conclusión de que se cumplió con la meta planteada. El estudio realizado permitió identificar que la Generación de Entornos Virtuales es un area de pocas clasificaciones establecidas, lo cual es necesario para su mejor desarrollo y comprensión. También se identificó que existen pocas publicaiones de sistemas de generación en *tiempo real*, lo cual demuestra que es un area de trabajo en la cual se puede continuar trabajando.

El sistema resultante del presente trabajo constituye una novedosa solución, genérica tanto por su aquitectura así como por la nueva estrategia de generación de entornos que encapsula(Losas Dinámicas). Un sistema multitécnica, independiente del motor de visualización e inteligente resume sus potencialidades. Los resultados obtenidos trascienden el contexto del proyecto HDSRV, se aporta una solución genérica, flexible y con nuevos enfoques de generación de entornos infinitos no vistos en otros sistemas de generación.

A modo de recomendación, se cree que existen varios aspectos a considerar para lograr mejores resultados en el futuro. Estos son:

- Incorporación de nuevas estrategias de generación con vista a lograr la generación de una gama más amplia de entornos virtuales.
- Integración de las técnicas de optimización de los motores integrados con el Generador, para lograr resultados en *tiempo real* en entornos más complejos.

- Creación de un ficheros más robusto que permita la configuración del generador con vista a la generación de otros entornos. El propuesto por el sistema Scaper es un buen referente.

- Rediseño del módulo con vista a crear una biblioteca de generación de entornos virtuales en *tiempo real*.

A. Glosario de términos

Términos en inglés:

***Frustum*:** Volumen formado por los 6 planos que definen los límites de la cámara: cercano (*near*), lejano (*far*), izquierdo (*left*), derecho (*right*), arriba (*top*) y abajo (*bottom*).

***Frustum culling*:** Técnica de selección de objetos visibles determinando si están o no en el interior de *frustum* de la cámara.

***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 Inteligencia Artificial, dibuja, 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.

Frustum Filling: Llenado de Volumen a Visión.

Términos en español:

Fotogramas: (*Frames*) Cada una de las imágenes que se muestran como proceso final del renderizado de una escena 3D a imagen 2D. La secuencia de estas imágenes conforma la parte visual de un videojuego. La velocidad

con que se muestran es conocida como *frames* por segundo, especificando la cantidad de imágenes mostradas por cada segundo de aplicación.

Motor gráfico: Ver *graphic engine*.

Realidad Virtual: Representación de escenas u objetos producidos por un sistema informático, dando la sensación de su existencia real.

SceneToolKit: (STK) Alias del paquete de herramientas desarrollado por el proyecto, formada por *scene*, escena, y *toolkit*, "paquete de herramientas". El nombre completo es "Herramientas 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: (*Real time*) En gráficos por computadoras, las aplicaciones en tiempo real son aquellas que generan los fotogramas a medida que son necesarios durante la ejecución de la aplicación. El tiempo de respuesta de la aplicación debe ser lo suficientemente corto como para que la percepción del fenómeno se corresponda con fidelidad al sistema real.

Videojuego: Programa informático 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.

Videojuego Terapéutico: Videojuego creado con vista a darle soporte a un tratamiento de una enfermedad. En estos casos se puden encontrar juegos para el tratamiento de las fobias.

Contenido : Unidad básica de un entorno generado. Incluye a todas las geometrías que conforman a un objeto con un significado según la topología del entorno.

Agentes Inteligentes: entidad que percibe y actúa sobre un entorno

Coherencia Estructural: equilibrio y permanencia de componentes de un entorno virtual al pasar el tiempo.

B. Glosario de abreviaturas

FPS: *Frames* por segundo.

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

STK: SceneToolKit.

UCI: Universidad de las Ciencias Informáticas.

Referencias

[Ada02] Adams, David: *Automatic Generation of Dungeons for Computer Games*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.1445&rep=rep1&type=pdf, may 2002.

[Boo] Boo, Javi: *Sistemas Gráficos Interactivos (SGI), capitulo 2 Introducción a la Realidad Virtual*. Informe técnico.

[CGCA09] Correa, Omar, Orlay García, Yesy Cedeño y Leoder Alemañy: *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.

[Com08a] Community, Ogre. http://www.ogre3d.org/wiki/index.php/, Octubre 2008.

[Com08b] Community, Ogre: *Documentation Architecture*. http://www.ogre3d.org/wiki/index.php/Documentation_Architecture, Octubre 2008.

[Com08c] Company, Google: *GoogleEath3D*. http://www.ge3d.com/, Octubre 2008.

[Cor08a] Corporation, Blizzard. http://www.blizzard.com/war3x/, January 2008.

[Cor08b] Corpotation, Presagis. http://www.presagis.com/products/content_creation, January 2008.

[DME07] Darken, R., P. McDowell y E.Johnson: *Expeditious Modelling of Virtual Urban Environments with Geospatial L-systems.* COMPUTER GRAPHICS, 26(4):769–782, March 2007.

[Dym09] Dymchenko, Lev: *Polygonal Technology: Weaknesses and Alternative.* http://ixbtlabs.com/articles/virtualrayengine/index.html, February 2009.

[F.S07] F.Strugar: *Advantage Terrain Library.* http://www.advantageterrain.com/, 2007.

[GPSG03] GreuterS, Stefan, J. Parker, N. Stewart y L. Geoff: *Real-time procedural generation of pseudo infinite cities.* Informe técnico, 2003.

[GS04] Greuter, Stefan y Nigel Stewart: *Beyond the horizon.Computer generated, three-dimensional, infinite virtual worlds without repetition in real-time.* Image Text and Sound Conference, Australia, 2004.

[Haa99] Haahr, Mads: *Introduction to randomness and random number.* Informe técnico, 1999.

[HBW06] Hahn, E., P. Bose y A. Whitehead: *Persistent Realtime Building Interior Generation.* ACM Siggraph, páginas 179–186, March 2006.

[Inc08] Inc, Linden Research: *SecondLife.* http://secondlife.com/, Octubre 2008.

[Inn08] InnovativeFSP: *FlaySimulator.* http://www.innovativefsp.com/, Octubre 2008.

[Int] Intertaiment, Red Storm, January.

[LWW08] Lipp, Markus, Peter Wonka y Michael Wimmer: *Interactive Visual Editing of Grammars for Procedural Architecture.* Informe técnico, 2008.

[Mad09] Madrigal, Omar Correa: *Generación de Entornos Virtuales en Tiempo Real, una solución eficiente y ajustable al tratamiento de fobias.* VI Congreso Internacional de Tecnologías, Contenidos Multimedia y Realidad Virtual, en XIII Convención y Expo Internacional Informática, 2009.

[Mau03] Maurice, Danaher: *DYNAMIC LANDSCAPE GENERATION USING PAGE MANAGEMENT.* http://wscg.zcu.cz/wscg2002/papers_2002/d97.pdf, 2003.

[MH02] Moeller, Akenine y Eric Haines: *Real-Time Rendering.* AK Peters, 2002.

[MWH+06] Muller, P., P. Wonka, S. Haegler, A. Ulmer y L. Van Gool: *Procedural Modeling of Buildings.* ACM SIGGRAPH, 2006.

[nez05] nez, J. Maroto Iba: *Metodología para la Generación de Entornos Virtuales distribuidos y su aplicación a simuladores de conducción.* Tesis de Doctorado, 2005.

[Noc05] Nocke, Frank: *Scaper.Real-Time Generation of Infinite Environment.* Tesis de Licenciatura, 2005.

[Pub03] Pubb, Greg: *Terrain Engine usin C++ y Directx 9.* Charles River Media,Massachusetts, 2003.

[Rav04] Ravin, Steve: *AI Game Programming Wisdom 2*. 2004.

[S.I99] S.Ince: *Automatic Dynamic Content Generation for Computer Games*. Informe técnico, 1999.

[SJ03] S.Greuter y J.Parker: *Undiscovered Worlds - Towards a Framework for Real-Time Procedural World Generation*. 2003.

[Sta04] Stainless, Shaw Shoemaket: *AI GAme Programing. Wisdom II*, capítulo Random Map Generation for Strategy Games, páginas 405–412. 405–412, 2004.

[Vis] Visualization, Interactive Date: *Interactive Data Visualisations.Speedtree RT*.

[Worc] Worlds, Binary: *Decensor Engine*. http://www.binaryworlds.com/products.html, c.

Fecha de impresión: 13 de septiembre de 2010

Generado con LaTeX