CEIG'08, Barcelona, Sept. 3–5 (2008)
L. Matey and J. C. Torres (Editors)

Técnicas para visualización de lluvia en entornos virtuales

Anna Puig-Centelles, Oscar Ripolles, Miguel Chover

Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I, Castellón, España
{apuig,oripolle,chover}@uji.es

Abstract

Los entornos virtuales incluyen a menudo lluvia y otros fenómenos atmosféricos que dan mayor realismo a la escena aunque son difíciles de simular en tiempo real. Dentro del campo de la informática gráfica se han presentado algunas soluciones que ofrecen sistemas de lluvia realistas pero costosos. Nuestra propuesta consiste en desarrollar una solución que facilite la creación de escenas con lluvia y reduzca el coste de los sistemas de partículas, mientras se mantiene la apariencia realista de la lluvia. Este objetivo se alcanza, por una parte, mediante la definición de áreas de lluvia, y por otra, gestionando adecuadamente los sistemas de partículas que creamos dentro de ellas. Incluimos un enfoque multirresolución para adaptar el número de partículas, su ubicación y tamaño a la posición del observador y a su distancia respecto a las partículas. La estructura que proponemos está integrada completamente en la GPU, ofreciendo una solución sencilla, eficiente y fácil de integrar en aplicaciones existentes. Además, en este trabajo se analizan las propiedades físicas de la lluvia y sus características se incorporan a la solución final.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Virtual reality; I.3.3 [Computer Graphics]: Display algorithms

1. Introducción

Las ciencias atmosféricas han estudiado en profundidad el fenómeno de la lluvia [MP48], [Ras95]. Estos estudios incluyen un análisis de la apariencia de una gota de lluvia a través de sus propiedades físicas, como son su distribución de forma, tamaño y velocidad. Además, es posible encontrar otras investigaciones meteorológicas que examinan las propiedades visuales de la lluvia: densidad, tamaño, velocidad de las gotas y dirección de su movimiento [SB00].

La lluvia es un fenómeno natural muy complejo. Dentro del campo de la informática gráfica es posible encontrar diversos trabajos que tratan de ofrecer una solución a la representación de sus componentes individuales, como gotas de lluvia cayendo desde el cielo [GN06] [WW04] o como gotas de agua interaccionando con superficies [WMT05].

El trabajo que presentamos se centra en la representación de gotas de lluvia en tiempo real. La mayor parte de las soluciones desarrolladas puede clasificarse en dos grupos: basadas en imágenes y basadas en sistemas de partículas. En general, los autores se han centrado más en modelos de simulación de lluvia basados en sistemas de partículas, ya que son capaces de ofrecer una mayor eficiencia. Además, algunas de estas soluciones intentan aprovechar las posibilidades de las GPUs actuales, que utilizan una arquitectura programable que permite incorporar a la *pipeline* gráfica *vertex*, *geometry* y *pixel shaders*. Sin embargo, aún existe un vacío en la creación de un sistema completo para la visualización de la lluvia en tiempo real en entornos complejos, debido al elevado coste computacional de las soluciones existentes y al hecho de que estas soluciones fallen en entornos interactivos en los que el usuario se mueve frecuentemente.

El uso de técnicas de nivel de detalle aumenta la eficiencia del proceso de visualización de un modelo, disminuyendo la carga de trabajo en distintos niveles de la *pipeline* gráfica. El modelado multirresolución se ha aplicado con éxito para solucionar problemas en muchas áreas [PS97] y existe una cantidad considerable de bibliografía disponible [LRC*03]. Aun así, no existe mucho trabajo de investigación sobre modelos multirresolución para sistemas de partículas. Podríamos destacar el modelo presentado en [OFL01] donde se presenta una jerarquía de partículas y se propone el uso

de las propiedades físicas para cambiar de un nivel de detalle a otro cuando fuera necesario.

El trabajo presentado en este artículo se propone como solución para aligerar el coste producido al gestionar y visualizar sistemas de partículas. Uno de los principales objetivos es ofrecer un sistema de lluvia adecuado para las escenas en las que el usuario se mueve continuamente, como los entornos de realidad virtual o los juegos de ordenador. Para conseguir este objetivo proponemos un método para la creación automática de entornos con lluvia a través de la definición de áreas de lluvia y la gestión adecuada de sistemas de partículas creados dentro de ellas. Además, incluimos el concepto de multirresolución para ajustar el tamaño y el número de partículas a las condiciones de la escena. La solución presentada incluye las siguientes características:

- Análisis de requisitos físicos para simular lluvia realista.
- Definición y manejo de zonas de lluvia, así como métodos de transición suave entre zonas con distintas condiciones de lluvia.
- Sistema de partículas multirresolución para la lluvia.
- Implementación en GPU, aprovechando *vertex shaders*, *pixel shaders* y los más recientes *geometry shaders*.

Este artículo presenta la siguiente estructura: la sección 2 contiene el estado del arte del trabajo llevado a cabo previamente y presenta una caracterización de los modelos de lluvia existentes. La sección 3 analiza las físicas de la lluvia y presenta el marco de desarrollo del modelo de lluvia con las principales características de nuestro método, mientras que la sección 4 muestra la implementación del nivel de detalle. La sección 5 presenta nuestros resultados y los compara con otras soluciones y la sección 6 contiene comentarios sobre resultados y futuras líneas de trabajo.

2. Estado del arte

Es posible encontrar en la bibliografía distintos métodos para representar lluvia dentro del campo de la informática gráfica. En la mayoría de casos, se asume que las gotas tienen formas simples, como rectángulos o elipses, y que la luminosidad de cada rayo es constante.

La lluvia se visualiza tradicionalmente bien como un sistema de partículas o bien como geometría centrada en la cámara por medio de texturas deslizantes. Además, también es interesante mencionar algoritmos que añaden lluvia a un video con objetos en movimiento, que nosotros denominamos soluciones basadas en la imagen.

2.1. Soluciones basadas en la imagen

La simulación fotorrealista de la lluvia tiene en cuenta que la iluminación, los efectos del punto de vista, la reflexión y la refracción producen patrones de luminosidad complejos. El trabajo presentado en [GN06] afirma que el patrón de luminosidad de un rayo de lluvia incluye típicamente motas, múltiples puntos de luz borrosos y perfiles de luminosidad curvados. Estos autores han desarrollado un algoritmo basado en imágenes que utiliza una base de datos de gotas para añadir lluvia a una única imagen o a un video con objetos en movimiento.

También es interesante mencionar el artículo [SW03], que simula lluvia en videos. Tras extraer lluvia de un video, aplican la lluvia extraída a un fondo diferente o lo vuelven a integrar en el video al que habían quitado la lluvia. No obstante, esta técnica se limita a escenas estáticas con puntos de vista fijos. De forma similar, podemos mencionar el trabajo presentado en [GN04], donde se estudia cómo añadir o eliminar lluvia de los videos.

2.2. Texturas deslizantes

La idea básica de estos modelos consiste en usar una textura que cubra toda la escena y se deslice continuamente siguiendo la dirección de la lluvia que cae. Así, esta textura se desplaza permanentemente de arriba a abajo. Algunos autores han desarrollado soluciones más complejas que incluyen varias capas de lluvia para simular lluvia a diferentes distancias del observador. Las texturas dinámicas son secuencias de imágenes que se pueden aplicar para simular las olas del mar, el humo, el follaje, tornados, etc. Sin embargo, muestran ciertas propiedades estacionarias en el tiempo [SDW01], limitando el realismo de las escenas.

En [WW04] se presenta una técnica para visualizar precipitación de manera realista y eficiente en escenas con posiciones de cámara en movimiento. Mapean texturas en un doble cono y las trasladan y alargan modificando las coordenadas de textura. Esta técnica es rápida, pero no permite la interacción entre la lluvia y el entorno. Así, este método no consigue crear una impresión de lluvia realmente convincente debido a que la lluvia no interacciona con la iluminación de la escena con precisión. Además, los resultados pueden parecer faltos de sensación de profundidad.

2.3. Sistemas de partículas

Este método se ha usado para simular ciertos fenómenos difusos que son difíciles de visualizar con técnicas de representación convencionales. Tradicionalmente, este ha sido el enfoque elegido para la visualización de lluvia en tiempo real; sin embargo, los sistemas de partículas suelen ser costosos, especialmente si queremos representar lluvia intensa. En consecuencia, los autores han orientado sus esfuerzos hacia la simplificación de estos sistemas.

Uno de los primeros autores en aplicar estas técnicas fue K. Kusamoto [KTT01], combinado un sistema de partículas con un modelo de movimiento físico de la lluvia sin aceleración en la GPU ni interacción con la luz. Posteriormente,

Feng et al. [FTDC06] incorporaron la detección de colisiones y un subsistema para simular las gotas de lluvia que salpican tras la colisión. Al representar escenas, se procesan de manera diferente según la distancia al espectador, aplicando efectos de profundidad de campo (DOF), añadiendo una fina niebla y evitando visualizar gotas de lluvia remotas.

El trabajo presentado en [WLF*06] propone una técnica que consiste en dos partes: el análisis previo de imágenes de vídeos de lluvia y la síntesis de lluvia basada en partículas en tiempo real, transfiriendo detalles del video de lluvia real a un sistema de partículas. Este método es económico a la vez que capaz de representar lluvia realista en tiempo real. Consideran el color de la escena, el desenfoque de la escena, la niebla y las salpicaduras hechas a mano o extraídas de un vídeo.

Desde una perspectiva diferente, un trabajo de N. Tatarchuk [Tat06] presenta un entorno de lluvia intensa, incluyendo gotas de lluvia, destellos, reflejos y detalles como las huellas de neumático en los charcos de la calle. Proporcionan un alto nivel de control artístico para conseguir el aspecto final deseado. Ofrecen un sistema híbrido de un enfoque basado en imágenes que compone múltiples capas para la lluvia general y efectos basados en partículas para las gotas de lluvia individuales y las salpicaduras. De todas maneras, este método requiere la utilización de 300 *shaders* individuales dedicados exclusivamente a la lluvia y la potente tarjeta gráfica Radeon X1800. Además, presenta una importante restricción ya que la cámara no puede cambiar su dirección.

En el artículo [RJG06], Pierre Rousseau et al. proponen un método de simulación realista de lluvia en tiempo real que simula la refracción de la escena en el interior de una gota de lluvia. La técnica que presentan intenta generar resultados precisos sin un alto coste computacional. Así, la escena se captura en una textura que se distorsiona de acuerdo con las propiedades ópticas de las gotas de lluvia. Su método también tiene en cuenta la persistencia retiniana en la que gotas de lluvia parecen rayas y la interacción con fuentes de luz.

Más recientemente, en [Tar], un trabajo desarrollado por S. Tariq presenta una aplicación de lluvia realista programada en DirectX10 y HLSL. Describen un método de sistema de partículas para animar y representar rayas de lluvia que se ejecuta enteramente en la GPU, usando nuevas funciones introducidas con Direct3D10. Las partículas de lluvia se animan en el tiempo utilizando las posibilidades de reutilizar el flujo de datos de la GPU y en cada imagen se expanden en impostores por medio del *geometry shader*. Finalmente, la representación de las partículas de lluvia emplea una biblioteca de texturas almacenadas en un vector de texturas que codifica la apariencia de diferentes gotas de lluvia bajo distintos puntos de vista y direcciones de iluminación. Esta biblioteca de texturas la obtienen del trabajo presentado en [GN06].

La mayor limitación de este tipo de métodos es que no son adecuados para escenarios en los que el usuario se mueve rápidamente, ya que ajustar el sistema de partículas a las nuevas condiciones de la cámara es generalmente demasiado costoso.

3. Nuestro modelo de lluvia

Como se puede observar en la sección previa, la representación de lluvia en tiempo real ha sido abordada desde diferentes puntos de vista, aunque la mayor parte de las soluciones se han basado en sistemas de partículas. El mayor inconveniente de utilizar sistemas de partículas para simular lluvia es la gran cantidad de partículas que debemos trasladar y visualizar para dar una sensación realista.

Uno de los principales objetivos de nuestro modelo es crear de forma automática entornos de lluvia mediante la generación de áreas de lluvia donde se simulen miles de gotas en tiempo real. Dentro de cada área, simularemos la lluvia con un sistema de partículas. Además, utilizaremos técnicas multirresolución que mejoren el rendimiento de la solución.

El funcionamiento de nuestra propuesta se basa en dos aspectos diferentes:

- El uso de distintos comportamientos de nuestra solución dependiendo de la ubicación del observador.
- La gestión multirresolución de las partículas.

Antes de adentrarnos en la simulación de lluvia en tiempo real es importante estudiar el modo en que la lluvia se produce y se precipita en la naturaleza.

3.1. Propiedades físicas de la lluvia

En primer lugar, cabe analizar por qué las gotas de lluvia pequeñas son casi esféricas y las más grandes se achatan por abajo adoptando una forma elipsoidal. Esta forma se debe al equilibrio entre la tensión superficial y la presión aerodinámica. La primera intenta minimizar la superfície de contacto entre el aire y la gota de lluvia y la segunda estira la gota horizontalmente. Las gotas pequeñas de hasta 0,5 mm son esféricas y las mayores se vuelven elipsoidales. En estudios previos se descubrió que el tamaño de la gota de lluvia sigue una distribución exponencial [Bes50]. Existe una velocidad específica asociada a cada gota. La velocidad de caída de una gota de lluvia depende de su radio, mientras que la velocidad terminal depende de la gravedad y la fricción. A nivel del suelo se ha medido que una gota esférica con un radio de 0,1mm a 0,45 mm tiene una velocidad terminal de 0,72 m/s a 3,67 m/s. Por su parte, una gota elipsoidal con un radio de 0,5 mm a 4,0 mm tiene una velocidad terminal de 4,0 m/s a 9,23m/s.

También es importante subrayar que se tiene la idea común pero equivocada de que una gota de lluvia tiene forma de lágrima como en la Figura 1(a), pero esto es inexacto

(a) Lluvia en forma de lágrima. (b) Lluvia en forma de rayas.

Figure 1: *Apariencia de la lluvia.*

debido a la percepción real que obtenemos al mirar la lluvia en la Figura 1(b). El ojo humano o una cámara a menudo perciben las gotas de lluvia como rayas. La retina del ojo humano recibe imágenes en tiempo continuo, y el obturador de una cámara recibe el flujo de imágenes en tiempo discreto. Al observar la lluvia, una gota sigue cayendo durante un lapso de tiempo que no es percibido ni por la retina ni por el obturador. Dicho efecto se denomina *persistencia retiniana* en el caso del ojo humano o *desenfoque de movimiento* para la cámara. Así, para representar este efecto de forma realista, a las partículas de lluvia se les da forma de rayas verticales, cuyos píxeles reciben contribuciones de las sucesivas posiciones de la gota.

3.2. Simulación de áreas de lluvia

Los entornos de lluvia del mundo real incluyen áreas donde llueve, así como áreas en las que no llueve. De esta manera, aunque esté lloviendo en un área muy amplia, debemos suponer que siempre podremos encontrar alrededor de ella áreas en las que no llueve. Tradicionalmente, la simulación de lluvia no ha considerado esta cuestión y los sistemas de lluvia no producen transiciones suaves entre áreas con diferentes condiciones de lluvia.

La solución que presentamos pretende establecer con bastante precisión dónde está lloviendo y dónde no dentro de un escenario dado. Además, también queremos manejar transiciones para crear un efecto de lluvia visualmente realista. Un aspecto importante de este realismo es la posibilidad de mirar hacia un área de lluvia lejana desde un lugar en el que no llueve. Hemos considerado que un área de lluvia se simula como una elipse con dos radios inicialmente dados. Esta elipse puede definirse con la siguiente función implícita:

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}=1 \qquad (1)$$

considerando que la elipse tiene su centro en x_0, y_0 y radios a, b, siendo $a, b \in \Re$ y $a > b$.

Además del área de lluvia, será también necesario crear y gestionar un *contenedor de lluvia*. Este contenedor engloba todo el conjunto de gotas de lluvia incluidas en nuestro sistema de partículas. En nuestro caso, el contenedor de lluvia tendrá forma de cilindro de base elipsoidal, que permite un mejor ajuste al campo de visión del observador. Con el objetivo de conseguir un mejor rendimiento y para obtener una distribución de partículas más adecuada para el nivel de detalle, decidimos omitir la parte trasera del cilindro. En la Figura 2 mostramos una imagen a vista de pájaro de nuestro contenedor de lluvia semicilíndrico, donde se puede observar cómo se sitúan en el escenario el observador, el *frustum* y el contenedor de lluvia.

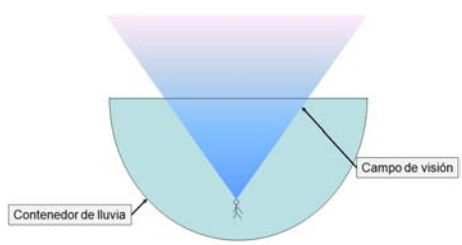


Figure 2: *Lluvia cercana.*

Nuestra solución propone la definición de un contenedor de lluvia más grande que los de soluciones anteriores [RJG06] [Tar]. El coste de mantenimiento de los sistemas de partículas de estas soluciones hace imposible el manejo en tiempo real de contenedores de lluvia de mayor tamaño. La gestión de un contenedor de estas dimensiones implicaría la creación y actualización de una cantidad demasiado elevada de partículas.

Para superar esta limitación y poder utilizar un contendor de mayor tamaño, nuestra solución incorpora técnicas de

nivel de detalle para modificar el tamaño y ubicación de las partículas mientras se conserva la apariencia realista de la lluvia. De esta manera, la combinación de estas técnicas multirresolución con la forma escogida para el contenedor ofrece una solución que es más adecuada para responder de forma eficiente a los cambios en la posición y dirección del *frustum* del usuario.

El usuario podría interactuar con un área de lluvia de dos modos diferentes:

1. El usuario está dentro del área de lluvia, completamente rodeado de gotas de lluvia. Hemos denominado esta situación lluvia cercana.
2. El usuario está fuera del área de lluvia, mirándola de un lugar lejano o alejándose del área. En estos casos utilizamos el modelo de lluvia lejana.

Estas dos posibilidades se muestran en la Figura 3. Es importante señalar que el contenedor de lluvia presenta distintos comportamientos dependiendo de si utilizamos el enfoque de lluvia cercana o de lluvia lejana. Así, la imagen de la izquierda hace referencia a la lluvia cercana mientras que las otras dos imágenes muestran dos posibles casos en los que debemos utilizar la lluvia lejana.

Lluvia cercana

Este caso sucede cuando el observador se encuentra dentro del área de lluvia. Para comprobar si el observador está dentro de esta área, usamos la Ecuación 1. Podemos verificar la ubicación del observador comprobando si el miembro izquierdo de la ecuación es inferior a 1. En ese caso, el observador está dentro de la elipse.

El contenedor de lluvia se centra inicialmente en el observador, tal y como se muestra en la Figura 2. Seguidamente, la ubicación y orientación de este contenedor se actualiza continuamente para seguir los movimientos del usuario. Así pues, si el usuario se desplaza el contenedor se desplaza con él y si el usuario cambia la dirección de su vista el contenedor debe reorientarse también. No obstante, hemos dado al usuario cierto margen de movimiento para permitirle hacer pequeños desplazamientos y cambios en la dirección de visión sin alterar la posición de todo el contenedor de lluvia. De esta manera, si el usuario sobrepasa los límites de estos márgenes, el contenedor de lluvia se reubicará para seguir el movimiento del usuario y las nuevas condiciones de visión. De forma más concreta, si el usuario realiza cambios en su orientación de menos de 90^{o} no será necesario reorientar el contenedor ya que la percepción de lluvia se mantiene. Esta característica se debe a la forma del contenedor y supone una ventaja respecto a soluciones anteriores.

Lluvia lejana

La lluvia lejana se refiere a un escenario donde el observador se sitúa lejos del área de lluvia. Esta situación sucede cuando la cámara está en un área en la que no llueve y puede mirar hacia el área donde realmente está lloviendo.

Para comprobar si el usuario puede ver esta área de lluvia, elegimos de entre el perímetro de la elipse el punto más cercano al observador y luego hacemos un simple test de este punto contra el *frustum*. Comprobamos en qué lado de los planos está situado el punto para saber si está dentro o fuera del *frustum*. Si este punto pasa la prueba del *frustum*, nuestro modelo de lluvia debe reaccionar de forma adecuada para representar correctamente el sistema de partículas.

El primer paso es situar el contenedor de lluvia en el perímetro del área de lluvia elipsoidal. Dentro de este perímetro, el sistema de lluvia escogerá el punto más cercano al observador para centrar allí el contenedor de lluvia. Por otra parte, deberemos modificar el tamaño del contenedor de lluvia para cubrir toda la parte de la elipse que esté mirando el observador. Para ello, tendremos en cuenta las dos rectas laterales que definen el frustum para comprobar cuál es la relación entre estas rectas y la elipse que define el área de lluvia. Así pues, comprobando las intersecciones entre estas dos rectas y la elipse podremos saber si el observador puede ver toda el área de lluvia o solo una parte. En el caso que el observador solo pueda ver una parte, el mismo test nos indica el tamaño de esta parte para ajustar el tamaño del contendor correctamente. La Figura 3 muestra dos posibles adaptaciones del contenedor de lluvia lejana según las características de la vista del observador. De forma similar, la altura del contenedor debe expandirse para simular las gotas de lluvia cayendo del cielo.

En segundo lugar, la cantidad de partículas a trasladar y visualizar, así como su tamaño, se adaptan en función de la distancia al observador.

La Figura 4 muestra una imagen de la lluvia lejana que acabamos de presentar, obsérvese como en la parte del suelocercana al ususario no cae lluvia. Una parte del sistema de partículas ha sido representada en color rojo para presentar de una forma más clara cómo la lluvia está alejada del usuario y cómo las partículas cubren toda la zona visible del área de lluvia. Este es sólo un ejemplo que muestra lo que vería el usuario desde fuera del área de lluvia. Se podría simular una lluvia más intensa aumentando el número de partículas, así como su tamaño.

Transiciones entre las dos situaciones de lluvia

Dentro de la solución propuesta para la simulación de las partículas de lluvia es importante controlar de forma adecuada las transiciones que experimenta el usuario al cambiar de lluvia cercana a lluvia lejana y viceversa.

La transición de lluvia cercana a lluvia lejana sucede cuando el usuario sale del área de lluvia. Nuestro sistema detecta este cambio cuando el usuario se sitúa sobre un punto del perímetro de la elipse que definía inicialmente al área

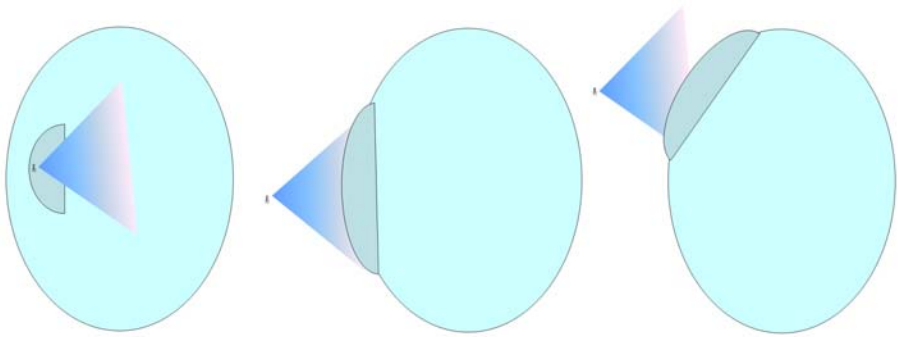


Figure 3: *Puntos de vista del usuario respecto al área de lluvia, mostrando varias posibilidades.*

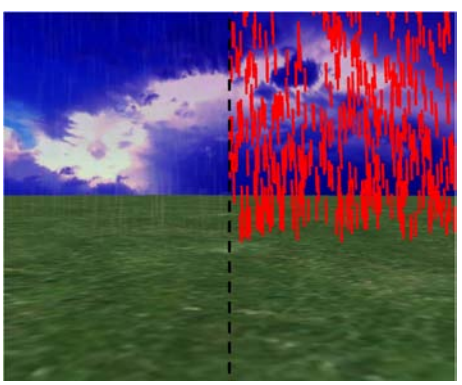


Figure 4: *Lluvia lejana.*

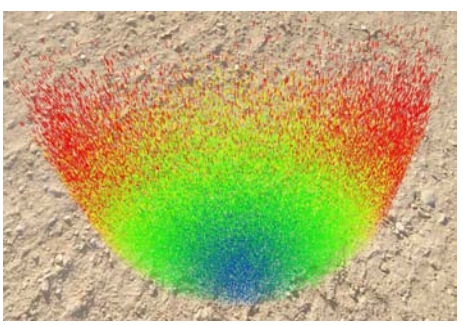


Figure 5: *Representación a vista de pájaro del tamaño y número de partículas.*

de lluvia. En este caso, el contenedor de lluvia permanecerá inmóvil en ese punto. Obtendremos una disminución progresiva de la cantidad de gotas de lluvia a medida que el usuario también se aleje del contenedor de lluvia que ya no sigue sus movimientos.

En cuanto a la transición de lluvia lejana a lluvia cercana, de nuevo utilizamos el perímetro del área de lluvia y controlaremos si es necesario realizar el cambio para mantener el realismo del sistema de lluvia. Como hemos explicado previamente, el tamaño del contenedor se adapta al tamaño del frustum del observador. De esta manera, a medida que el observador se acerque al área de lluvia este tamaño irá disminuyendo hasta que el usuario atraviese el perímetro del área de lluvia inicial. En ese momento se realizará el cambio entre los dos funcionamientos de nuestra solución.

3.3. Nivel de detalle para partículas de lluvia

Las partículas del interior del contenedor de lluvia deben modificarse siguiendo un patrón de nivel de detalle. La idea básica de nuestro algoritmo es adaptar la cantidad de partículas y su tamaño a la distancia al observador. Así, representamos más partículas y más pequeñas en las zonas que están más cerca del observador, y menos y más grandes en aquellas que están más lejos.

Se remite al lector a la Figura 5, que muestra una vista de pájaro de una escena en la que el observador se sitúa en el centro inferior de la figura. El color de las partículas sigue una gradación según la distancia al observador, empezando en azul las partículas más cercanas y terminando en rojo las más alejadas. Podemos ver cómo cuanto más cerca están las partículas del observador, mayor cantidad de partículas se visualizan. Con respecto al tamaño, también consideramos el criterio de la distancia para adaptar el tamaño de cada partícula con relación a la distancia. De este modo, las partículas situadas más cerca del espectador son más pequeñas que las que están más lejos.

Es importante comentar las funciones utilizadas para modificar la posición de las partículas y su tamaño. Cabe destacar que estas funciones son una de las partes básicas de nuestro modelo multirresolución. Inicialmente se decide un número de partículas que se mantendrá durante toda la simulación. La distribución de estas partículas no debe ser igual a lo largo de todo el contenedor. Como ya se ha comentado, deberemos representar más partículas en las zonas situadas más cerca del observador.

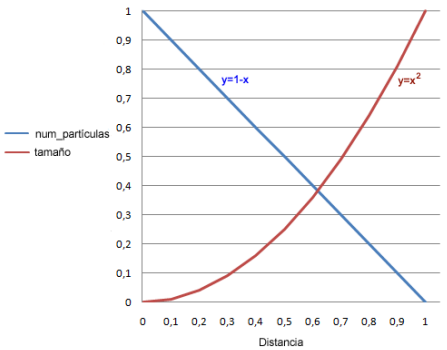


Figure 6: *Funciones utilizadas para la representación multirresolución.*

En el momento de inicializar las partículas, debemos mantener una proporción entre la distancia y el número de partículas. Esta relación sigue una distribución lineal (ver Figura 6), ya que necesitamos que ambos aumenten y disminuyan proporcionalmente, como se define en la Ecuación 2. Tal y como indica la ecuación, necesitamos que el número de partículas disminuya a medida que la distancia va aumentando.

En cambio, el tamaño de las partículas se actualiza continuamente. En este caso, la relación no es lineal sino cuadrática (ver Figura 6), ya que es necesario que el tamaño de las mismas aumente más despacio que la distancia. Esta distribución se define en la Ecuación 3. Esta función indica que el tamaño aumentará poco a poco al principio y más rápidamente al final, ya que únicamente queremos visualizar partículas grandes en aquellas zonas que incluyan pocas partículas (las zonas más alejadas el observador).

$$num\_particulas = 1 - distancia \quad (2)$$

$$tamaño = distancia^2 \quad (3)$$

4. Implementación

Uno de los principales objetivos en la implementación de nuestra estructura consiste en hacer un uso óptimo del hardware gráfico. De forma similar al modo en que funciona una CPU, la GPU de la tarjeta gráfica también tiene una *pipeline* que permite realizar diferentes tareas individuales muy rápidamente. La idea es inicializar los datos de entrada en la *pipeline* gráfica para mejorar la simulación utilizando las unidades programables de la GPU (*shaders*).

La estructura básica de nuestro modelo de lluvia es similar a la presentada en el trabajo de Pierre Rousseau et al. [RJG06] y Sarah Tariq [Tar]. Nuestra estructura tiene los siguientes pasos:

- Inicializamos un sistema de partículas para representar la lluvia en una zona especificada.
- Definimos una posición inicial para las partículas que se reutiliza cada vez que las partículas caen fuera de los límites y deben aparecer de nuevo.
- Actualizamos la ubicación de esta zona para seguir el movimiento del usuario.
- Utilizamos la *pipeline* gráfica dos veces para visualizar cada escena. La primera vez actualizamos la posición de las partículas al caer y la segunda representamos las partículas utilizando la GPU para expandir los puntos en *quads* orientados al usuario.

4.1. Inicialización de las partículas

En primer lugar, debemos llenar el contenedor de lluvia con gotas distribuidas siguiendo los criterios presentados en la sección anterior para tener una distribución lineal de las partículas en función de la distancia (ver Figura 5). Estas partículas se generan en la CPU para posteriormente cargarlas en la GPU. Toda la información sobre las gotas de lluvia se guarda en una estructura de datos: tamaño original, posición original, posición actual y velocidad. Estos registros se guardan directamente de la CPU a la GPU, evitando el uso de texturas como ocurría en modelos previos [RJG06].

Es importante señalar que el tamaño y la velocidad de las partículas deben ser diferentes de una partícula a otra para evitar que todas las partículas tengan el mismo aspecto y sigan exactamente la misma trayectoria. El tamaño se ajusta siguiendo la información sobre el tamaño de las gotas de lluvia reales. Este tamaño se utilizará como referencia a la hora de modificar el tamaño de cada partícula dentro de nuestro modelo multirresolución. Además, la velocidad se ajusta al tamaño de la partícula según la información dada en [Ros00], simulando el modo en que se comporta la lluvia en el mundo real.

4.2. Representación de las partículas

Como mencionamos al principio de esta sección, necesitamos pasar dos veces por la pipeline para conseguir cada imagen de la escena final.

En la primera pasada, el *vertex shader* actualiza las posiciones y reubica las partículas que se han salido de los límites. El *geometry shader* utiliza el *StreamOut* para almacenar las posiciones actualizadas en vectores de vértices. Estos vectores utilizarán una técnica de *pingpong*, que consiste en tener dos vectores, uno para leer las posiciones actuales y el otro para escribir y actualizarlas. Más tarde, una vez los cambios se han realizado, estos vectores son intercambiados.

En la segunda pasada por la *pipeline* es cuando realmente representamos las partículas de lluvia. El *vertex shader* es el estándar, el *geometry shader* transforma cada partícula en un *quad* orientado a la cámara y el *pixel shader* calcula el color final.

4.3. Animación de las partículas

También es importante analizar el modo en que las partículas evolucionan a medida que pasa el tiempo. Esta evolución afecta a la posición y al tamaño de las partículas.

En primer lugar, respecto a la posición de las gotas, el *vertex shader* de la primera pasada se encarga de desplazarlas y reubicarlas en nuevas posiciones en el caso de que caigan fuera de los límites. En este caso, la nueva posición se calcula considerando la posición original almacenada en la estructura de datos de la partícula y la posición actual del observador. En los casos en que el observador está lejos del área de lluvia, las gotas se reubican alrededor del perímetro de la zona de lluvia y su posición se modifica en horizontal y vertical para cubrir toda el área de lluvia que ve el observador.

En segundo lugar, es necesario ajustar del tamaño de las partículas dependiendo de la distancia al observador. Con este propósito, el *geometry shader* de la segunda pasada se encarga de calcular el tamaño adecuado antes de convertir la partícula en un *quad*. De nuevo, el método de lluvia lejana requiere una implementación ligeramente diferente, pues necesita tamaños mayores para que las partículas cubran toda el área de lluvia visible para el observador.

5. Resultados

Se ha estudiado la calidad visual obtenida, el rendimiento y el número de partículas que es capaz de visualizar en tiempo real. Nuestra solución se ha programado con HLSL y C++ en un sistema operativo Windows Vista, utilizando el reciente DirectX10, que incorpora muchas mejoras y permite utilizar *geometry shaders* y el sistema de flujo de salida hacia los recursos de memoria (*StreamOut*). Todas las pruebas se han realizado en un Pentium D 2,8 GHz. con 2 GB de RAM y una tarjeta gráfica nVidia GeForce 8800 GT.

Empezamos presentando una comparación visual entre nuestra solución y dos métodos de lluvia previamente desarrollados basados en sistemas de partículas: Pierre et al. [RJG06] y Sarah Tariq [Tar]. La Figura 7 ofrece tres instantáneas de estos métodos simulando un entorno de lluvia fuerte con una intensidad similar. Puede comprobarse cómo nuestro enfoque puede ofrecer una mejor apariencia de lluvia con una cantidad de partículas considerablemente menor, reduciendo este número hasta alrededor del 30 % en comparación con el método de Sarah Tariq y del 55 % en comparación con Pierre et al. Como se puede observar en la Tabla 1, el número de imágenes por segundo (fps) obtenido en nuestro método es notablemente mayor que en las otras soluciones. Es importante indicar que el cálculo de los *fps* se ha realizado exclusivamente visualizando las partículas de lluvia, sin incluir elementos en el escenario, interacciones con luces de fondo u otros efectos avanzados. Así, el modelo que presentamos es capaz de obtener un aspecto visual similar al de los otros métodos con un menor número de partículas y un mayor rendimiento.

6. Conclusiones

Se ha presentado un conjunto de técnicas para crear y visualizar eficientemente escenarios con lluvia realista. Para ello, se ha incluido un método de nivel de detalle para simular lluvia en un entorno en tiempo real. Nuestra técnica proporciona diferentes enfoques dependiendo de la relación entre la ubicación del usuario y la situación del área de lluvia.

Los resultados que hemos obtenido mejoran los de soluciones anteriores. Los tests realizados muestran que nuestro enfoque multirresolución puede ofrecer una alta calidad visual a la vez que disminuye el número de partículas a representar. Debemos señalar que el alto rendimiento de nuestro método sólo se alcanza utilizando *shaders* de GPU. Además, con nuestro método solucionamos una importante limitación de sistemas anteriores: los movimientos de cámara rápidos, gracias a que nuestro contenedor de lluvia es más grande y su forma se ajusta más al campo de visión del observador. Los sistemas de partículas de modelos anteriores no son adecuados para entornos de juegos donde el usuario hace movimientos rápidos y continuos.

Finalmente, hemos demostrado que nuestro método es capaz de manejar grandes áreas de lluvia que consistirían de cientos de miles de partículas en tiempo real. Este trabajo muestra que es posible incorporar un esquema multirresolución a sistemas de partículas para simular áreas de lluvia. La estructura que hemos propuesto podría combinarse fácilmente con métodos para simular salpicaduras de gotas de lluvia, interacción de la luz y otros efectos que se han presentado en otros estudios.

Agradecimientos

Este trabajo ha sido subvencionado por el Ministerio Español de Ciencia y Tecnología (beca TSI-2004-02940 y proyecto TIN2007-68066-C04-02) y por Bancaja(P1 1B2007-56). Agradecemos a Pierre Rousseau y Sarah Tariq su colaboración.

References

[Bes50] BEST A.: *The Size distribution of Raindrops*. Quart. J. Royal Meteor. Soc., 1950.

[FTDC06] FENG Z., TANG M., DONG J., CHOU S.: Real-time rain simulation. In *CSCWD 2005, LNCS 3865* (2006), pp. 626–635.

[GN04] GARG K., NAYAR S. K.: Detection and removal of rain from videos. *IEEE Conference on Computer Vision and Pattern Recognition 01* (2004), 528–535.

[GN06] GARG K., NAYAR S. K.: Photorealistic rendering of rain streaks. *ACM Transactions on Graphics 25*, 3 (2006), 996–1002.

[KTT01] KUSAMOTO K., TADAMURA K., TABUCHI Y.: A method for rendering realistic rain-fall animation with motion of view. *IPSJ SIG*, 106 (2001), 21–26.

Modelo	Pierre et al. [RJG06]	Sarah Tariq [Tar]	Nuestro modelo
Imágenes por segundo (fps)	89	95	187
Número de partículas	90.000	195.000	50.000

Table 1: *Comparación de resultados para un entorno de lluvia intensa.*

[LRC*03] LUEBKE D., REDDY M., COHEN J., VARSHNEY A., WATSON B., HUEBNER R.: *Level of Detail for 3D Graphics*. Morgan-Kaufmann, Inc., 2003.

[MP48] MARSHALL J. S., PALMER W. M.: *The distribution of raindrops with size*. Journal of Meteorology, 1948.

[OFL01] O'BRIEN D., FISHER S., LIN M.: Automatic simplification of particle system dynamics. *14th Conference on Computer Animation.* (2001), 210–257.

[PS97] PUPPO E., SCOPIGNO R.: Simplification, lod and multiresolution - principles and applications. In *Eurographics '97 Tutorial Notes* (1997).

[Ras95] RASMUSSEN R. M.: A review of theoretical and observational studies in cloud and precipitation physics. *Rev. Suppl. American Geophysical Union 33* (1995).

[RJG06] ROUSSEAU P., JOLIVET V., GHAZANFARPOUR D.: Realistic real-time rain rendering. *Computers & Graphics 30*, 4 (2006), 507–518.

[Ros00] ROSS O.Ñ.: *Optical remote sensing of rainfall micro-structures*. Tech. rep., Freien Universität Berlin, 2000.

[SB00] STRAUBE J., BURNETT E.: Simplified prediction of driving rain on buildings. In *In Proceedings of International Physics Conference* (2000), pp. 375–382.

[SDW01] SOATTO S., DORETTO G., WU Y.Ñ.: Dynamic textures. In *Int. Conference on Computer Vision* (2001), pp. 439–446.

[SW03] STANIK S., WERMAN M.: Simulation of rain in videos. In *Int. Journal of Computer Vision Texture* (2003), pp. 95–100.

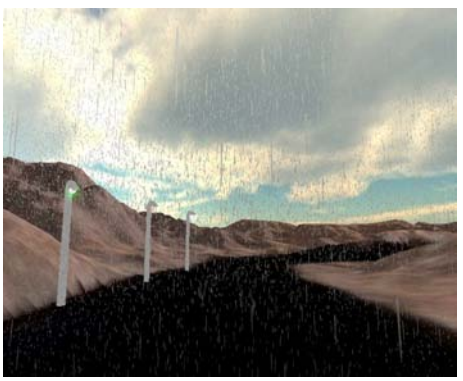
[Tar] TARIQ S.: Rain. nvidia 2007 whitepaper. http://developer.download.nvidia.com/whitepapers/2007/SDK10/RainSDKWhitePaper.pdf.

[Tat06] TATARCHUK N.: Artist-directable real-time rain rendering in city environments. In *ACM SIGGRAPH 2006 Courses* (2006), pp. 23–64.

[WLF*06] WANG L., LIN Z., FANG T., YANG X., YU X., KANG S.: Real-time rendering of realistic rain. In *ACM SIGGRAPH 2006 Sketches* (2006), p. 156.

[WMT05] WANG H., MUCHA P. J., TURK G.: Water drops on surfaces. *ACM Trans. Graph. 24*, 3 (2005).

[WW04] WANG N., WADE B.: Rendering falling rain and snow. In *ACM SIGGRAPH 2004 Sketches* (2004), p. 14.



(a) Modelo de lluvia de Rousseau et al. [RJG06] con 90.000 partículas.

(b) Modelo de lluvia de Sarah Tariq [Tar] con 150.000 partículas.

(c) Nuestro modelo de lluvia con 50.000 partículas.

Figure 7: *Comparación de la apariencia de modelos de lluvia.*