Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXIII, págs. 1695-1709 (artículo completo)
Graciela Bertolino, Mariano Cantero, Mario Storti y Federico Teruel (Eds.)
San Carlos de Bariloche, 23-26 Setiembre 2014

MODELADO PARA LA RENDERIZACION FOTO-REALISTA DE PAN

Rodrigo Baravalle[a], Leonardo Scandolo[b], Claudio Delrieux[c] y Cristian G. Bauza[d]

[a]*Laboratorio de Sistemas Dinámicos y Procesamiento de Señales, FCEIA, Universidad Nacional de Rosario, CIFASIS-CONICET, Ocampo y Esmeralda, S2000EZP Rosario, Argentina, baravalle@cifasis-conicet.gov.ar, http://www.cifasis-conicet.gov.ar/grupo4.html*

[b]*Departamento de Ciencias de la Computación, FCEIA, Universidad Nacional de Rosario, Pellegrini 250, 2000 Rosario, Argentina, leonardo@fceia.unr.edu.ar, http://web.fceia.unr.edu.ar/es/institucional/escuelas/118-departamento-ciencias-de-la-computacion-ecen.html*

[c]*Departamento de Ingeniería Eléctrica y de Computadoras, Universidad Nacional del Sur - IIIE-CONICET, Colón 80, 8000FTN Bahía Blanca, Argentina, cad@uns.edu.ar, http://www.ingelec.uns.edu.ar/*

[d]*Instituto de Investigación PLADEMA- Facultad de Ciencias Exactas - Universidad Nacional del Centro, Campus Universitario, Paraje Arroyo Seco, (B7001BBO) Tandil, Buenos Aires, Argentina crgarcia@exa.unicen.edu.ar, http://www.exa.unicen.edu.ar/es/d_investigacion/inst_plademe/index.html*

Palabras Clave: Direct Volume Rendering, Pan, Tiempo Real.

Resumen. El modelado foto-realístico de materiales con una estructura interna compleja presenta varios desafíos en Computación Gráfica. Específicamente, la miga del pan es un material translúcido cuya estructura es porosa, mostrando detalles en distintas escalas. Si se pretende obtener resultados foto-realistas, deben modelizarse fenómenos tales como sombras internas, oclusión, transmitancia y absorción. La solución típica en estos casos es la aplicación directa de la ecuación del rendering, es decir, utilizar iluminación global (ray tracing, path tracing). Sin embargo, estos métodos presentan un costo computacional elevado y necesitan una malla 3D detallada del material.

El estado del arte en renderización de materiales porosos utiliza un complejo procedimiento de captura donde la luz que es reflejada por el material es obtenida en distintos ángulos. Esa información es utilizada para reconstruir un modelo computacional del material. Si bien utilizando esta técnica es posible modelar alguna de las propiedades lumínicas deseadas, los costos computacionales asociados, el procedimiento de captura requerido y la baja variabilidad de la imagen resultante han provocado una dificultosa aplicación práctica del método.

En este trabajo proponemos el estudio e implementación en GPU de un modelo basado en *direct volume rendering* sobre un campo escalar representando la estructura de la miga de pan sin utilizar estructuras intermedias. Las imágenes obtenidas muestran resultados promisorios en tiempo real. La miga es representada a través de un campo escalar 3D, el cual es computado en dos pasos. El primero utiliza una generación a través de sistemas de partículas, y el segundo aplica sistemas dinámicos para evolucionar las partículas, emulando el proceso de leudado y cocción del pan.

1. INTRODUCCION

La apariencia de la miga de pan y otros materiales cocidos, como pizzas y budines, ha sido considerada un desafío para renderizar debido a la compleja interacción del material con la luz. Los costos computacionales asociados (almacenamiento de memoria y tiempo computacional) de estos procesos físicos hacen que el renderizado sea impráctico en aquellas áreas donde la interactividad sea mandatoria. El crecimiento exponencial en poder de cómputo, provocado mayormente por el diseño masivamente paralelo de las placas gráficas (Kenny Yeo, 2009; Harris y Luebke, 2006), ha hecho posible simular algunos fenómenos de la luz mencionados en tiempos computacionales aceptables, sin embargo el campo es aún objeto de estudio (Voglsam, 2013).

Estos materiales poseen una geometría observable la cual representa de por sí un desafío adicional. Estas estructuras porosas son el resultado de mecanismos complejos que involucran deformaciones físicas y reacciones químicas. La producción de pan presenta dos etapas: leudado y cocción. Durante el leudado, la levadura genera CO_2, produciendo burbujas en la masa (Shah et al., 1998). El proceso de cocción (Mondal y Datta, 2008) modifica estas burbujas de distintas maneras (Scanlon y Zghal, 2001), dándole al pan su estructura final. Se han hecho intentos de síntesis de esta estructura (Cho et al., 2007), pero el resultado es en realidad un proceso artístico. En este trabajo proponemos la utilización de sistemas dinámicos (Strogatz, 2001) aplicados a la evolución de sistemas de partículas (Reeves, 1983), utilizando un trabajo que los autores realizaron previamente (Baravalle et al., 2011), como un intento de imitar el leudado y la cocción mencionados. Los sistemas de ecuaciones diferenciales describen adecuadamente el comportamiento de procesos muy complejos, como el clima, la dinámica de fluídos (Stam, 1999), etc. Esta idea se utiliza en este trabajo para modelar el crecimiento de burbujas en el interior de panes. Este proceso arroja como resultado una estrucura similar a burbujas sobre un fluido, lo cual puede observarse en distintos panes. En otros trabajos se computa el valor de la textura en cada voxel del dominio (Perlin y Hoffert, 1989), por lo cual no se requiere memoria para alojar la textura 3D, sin embargo estos métodos requieren un enfoque estadístico que resulta inadecuado para capturar la distribución de las burbujas.

La estructura de datos utilizada en la representación de estas geometrías influye fuertemente el renderizado de las mismas. Si se utiliza una malla tradicional 3D, la misma debe computarse a partir de un campo escalar previamente construido, utilizando técnicas como *marching cubes* (Lorensen y Cline, 1987). Este proceso podría no resultar trivial debido a la estructura porosa del material. Representar la miga de pan como una superficie resulta también inadecuado, ya que existen estructuras (burbujas) sobre la misma, por lo cual soluciones típicas como utilizar funciones bidireccionales de distribución de reflectancia (BRDF) (Kurt y Edwards, 2009) no son factibles. Si bien existe una propuesta (Tong et al., 2005), ésta es demasiado compleja en su implementación, junto a un difícil proceso de captura asociado, la baja variabilidad en las imágenes resultantes y los costos computacionales del método, han hecho que el mismo no sea utilizado de manera masiva.

En este trabajo se propone la aplicación de Direct Volume Rendering (DVR) (Levoy, 1988; Kratz, 2006) sobre un campo escalar para renderizar el interior de distintos objetos sometidos a un proceso de cocción. El método de DVR consiste en lanzar rayos desde una cámara virtual hacia el campo escalar, acumulando distintas propiedades para cada pixel. El método no utiliza estructuras intermedias, simplificando el proceso de modelado. La forma exterior del material puede ser definida en tiempo real en la GPU, haciendo posible realizar cortes y deformaciones a la miga original. Además, la corteza del mismo se establece utilizando funciones de transferen-

cia (definición de distintas regiones en el espacio). Los resultados obtenidos son satisfactorios y se computan en tiempo real.

2. MATERIALES Y METODOS

2.1. Sistemas de Partículas

Los primeros trabajos en computación gráfica utilizaron la geometría Euclideana para describir figuras. Es decir, combinando puntos, rectas, planos y otras primitivas. Si bien esta metodología es útil para representar construcciones matemáticas simples, la misma presenta dificultades sobre todo en el modelado de objetos naturales. Estas limitaciones pueden ser superadas en muchos casos utilizando geometría fractal (Mandelbrot, 1982).

Distintas técnicas hicieron su aparición para superar estos problemas. Los sistemas de partículas (Reeves, 1983) suplieron la necesidad de trabajar con todo aquello que no tuviese superficie bien definida, como agua, fuego y humo (Cords, 2008; Paiva et al., 2009; Müller et al., 2003). Estos sistemas están compuestos por entidades denominadas *partículas*, las cuales modifican sus propiedades en el tiempo. Por ejemplo, se puede obtener una animación de fuegos artificiales definiendo un origen común en el espacio para todas las partículas, cambiando su posición en el tiempo siguiendo parábolas levemente diferentes para cada una de ellas. Otras propiedades como el color y el tamaño permiten definir fuego y humo. Las partículas pueden también afectarse mutuamente.

En un trabajo previo, (Baravalle et al., 2011), se emplearon distintos sistemas de partículas en la síntesis de texturas. Cada partícula tomaba una posición aleatoria en una imagen, y evolucionaba tratando de evitar otras partículas, compitiendo de esta forma por posiciones dentro de la misma. Se obtuvieron resultados adecuados para imágenes de madera y pinturas artísticas. Las funciones de crecimiento utilizadas fueron crecimiento vertical, aleatorio, horizontal, y diagonal. En este trabajo nos proponemos extender el dominio al espacio 3D, controlando el crecimiento de las partículas por medio de un sistema de ecuaciones diferenciales, imitando el proceso de leudado y cocción del pan. De esta forma se obtiene una textura 3D representando el material.

2.1.1. Algoritmo de modelado

El propósito de este algoritmo es producir una geometría la cual se renderizará posteriormente. Por lo tanto, en lugar de devolver el color de una posición específica, el algoritmo genera un campo escalar compuesto de 0s y 1s (0 si la posición contiene aire, 1 si la misma contiene masa). Esta representación resulta adecuada para ser renderizada utilizando DVR.

El sistema consta de un conjunto de partículas P,

$$P = \{p_1, ..., p_n\}, n \in \mathbb{N}, \tag{1}$$

una grilla $L_{N\times N\times N}, N \in \mathbb{N}$ (inicialmente $L_{xyz} = 1$) de masa y aire como fue descripto previamente, y otra grilla $L2_{N\times N\times N}$, (inicialmente $L2_{xyz} = -1$) de posiciones donde cada celda almacena un único entero que indica qué partícula es dueña de la misma (i si el elemento de la grilla pertenece al contorno o interior de la partícula i).

Cada elemento en P posee las siguientes propiedades:

$$p_i = \{O_i, C_i\}, 1 \leq i \leq n, \tag{2}$$

donde:

- $O_i = \{o_1, ..., o_{n_i}\}$: (Ocupadas) vector (conjunto) de posiciones ocupadas por la partícula en L.
- $C_i = \{c_1, ..., c_{m_i}\}$: (Contorno) vector (conjunto) de posiciones representando el *contorno* de la partícula en L.

El vector O representa las posiciones que serán afectadas por la partícula, y el contorno C se utiliza para asegurar que las partículas se eviten entre sí.

El algoritmo se describe, simplificadamente, en el siguiente pseudo-codigo:

Algoritmo 1 Algoritmo de modelado

```
t = 0                                              ▷ tiempo - iteración
P = []                                             ▷ partículas
L = matriz(MxMxM).valores_iniciales(1)             ▷ Geometría - iniciada a 1 (masa)
L2 = matriz(MxMxM).valores_iniciales(−1)           ▷ Dominio de cada partícula
for i ∈ [1, Cantidad_Particulas] do    ▷ Cada partícula toma una posición aleatoria en L
    x ← aleatorio, y ← aleatorio()
    O[i] ← [[x, y]]
    C[i] ← []
    for v ∈ vecindario(x, y) do
        C[i].agregar(v)
    P.agregar([O[i], C[i]])
for t ∈ [0, tiempo_max] do
    for i ∈ [1, Cantidad_Particulas] do
        if vacio? C[i] then
            morir()
        for h ∈ C[i] do
            C[i].eliminar(h)                       ▷ la posición ya fue explorada
            // Si la posición o el contorno pertenece a otra partícula, elegir otra posición
            // borde_libre chequea que el vecindario no este dominado por otra partícula
            if !(L2[h] > 0 && L2[h]! = i && borde_libre(separacion) then
                // Posición libre, ocupar
                L[h] ← 0                                   ▷ masa ->aire
                O[i].agregar(h)
                C[i].agregar(vecindario(h))
                L2.setear(vecindario(h), i)        ▷ Marcar posiciones en L2 como i
                L2.setear(h, i)                    ▷ turno de la particula i + 1...
```

Cuando $t = 0$, un conjunto de partículas iniciales toman posiciones aleatorias en la grilla. Para cada partícula, la posición elegida es la primera posición ocupada en O, además, el vecindario de O es agregado a C. Cada partícula evoluciona en un intento por extender sus posiciones ocupadas (O), marcando posiciones en L. Las posiciones se toman de C. Cuando una partícula ocupa una posición, la misma se elimina de C y se agrega a O. Luego el vecindario de esa posición se agrega a C (es decir las posiciones que rodean inmediatamente a la posición tomada). Las grillas se actualizan de la siguiente manera: L se setea a 0 en la posición y $L2$ se setea con el valor i en las posiciones que se agregan a C. Las partículas sólo pueden crecer si el valor encontrado en $L2$ no pertenece a otra partícula. El tamaño del vecindario es un parámetro que

define la distancia entre partículas. Si el vector C está vacío, la partícula *muere* ya que no puede continuar creciendo.

El algoritmo puede ser terminado en cualquier t deseado. El mismo puede finalizar su cómputo ante determinados eventos, por ejemplo, cuando todas las posiciones de $L2$ fueron tomadas por partículas, ya que no pueden realizarse progresos.

Variando el parámetro de distancia entre partículas (separación) se obtienen distintas estructuras (ver Fig. 1). Las imágenes muestran ejemplos 2D (para mayor claridad) de crecimiento aleatorio de partículas. La región blanca en las imágenes representa la masa restante luego del proceso.

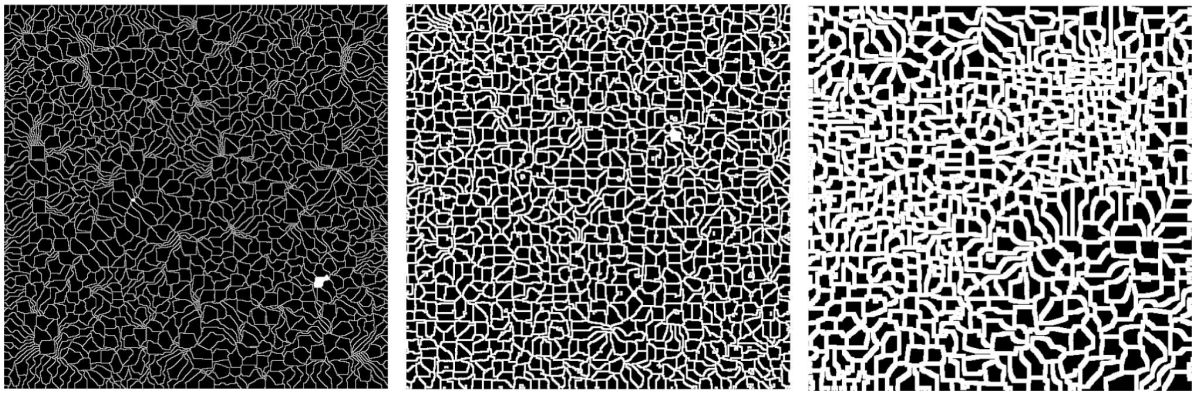

Figura 1: Diferentes separaciones entre partículas utilizando el parámetro separación. Izquierda: separación = 1, centro: separación = 2, derecha: separación = 4.

Finalmente, el algoritmo devuelve la grilla L, la cual será renderizada posteriormente. En la siguiente sección se explica la utilización de sistemas dinámicos en la evolución guiada de partículas.

2.2. Sistemas Dinámicos

Las ecuaciones diferenciales tienen como propósito tratar con la dificultad (o imposibilidad) de hallar soluciones analíticas en procesos dinámicos. En primer lugar, se define un modelo matemático del problema, del cual luego se obtienen las ecuaciones diferenciales asociadas. Los mismos se resuelven por medio de aproximaciones numéricas en cada instante de tiempo.

Los costos computacionales de estas soluciones dependen de la complejidad del problema y el número de ecuaciones del sistema. En este trabajo proponemos utilizar una sub-área de ecuaciones diferenciales llamada ecuaciones diferenciales ordinarias (ODE). En esta representación, el tiempo es tratado como la única variable independiente.

De manera general, las ODEs se representan utilizando el siguiente sistema de ecuaciones:

$$\begin{aligned} \dot{x_1} &= f_1(x_1, \ldots, x_n), \\ &\ldots \\ \dot{x_n} &= f_n(x_1, \ldots, x_n), \end{aligned} \tag{3}$$

donde $\dot{x_i}$ representa la derivada de x_i con respecto a t. Las variables x_i y las funciones f_i se definen de manera diferente para cada problema. En este caso, cada variable representa una coordenada cartesiana en el espacio, x_1 es x, x_2 es y y x_3 es z. El conjunto de f_i será definido tratando de capturar la estructura interna del pan. La siguiente sección muestra cómo estos sistemas pueden describir la evolución de los sistemas de partículas.

2.3. Evolución de sistemas de partículas utilizando sistemas dinámicos

La percepción humana puede detectar patrones en la estructura de la miga de pan. Distintas observaciones pueden realizarse sobre la distribución de las burbujas en la misma (ver Fig. 2). Primero, la forma de las burbujas cercanas a la corteza tiende a estirarse paralelamente a la misma. Esto es resultado de la acción de las elevadas temperaturas durante la cocción de la masa. También resulta evidente que la estructura completa es similar a un fluido con la forma de la corteza.

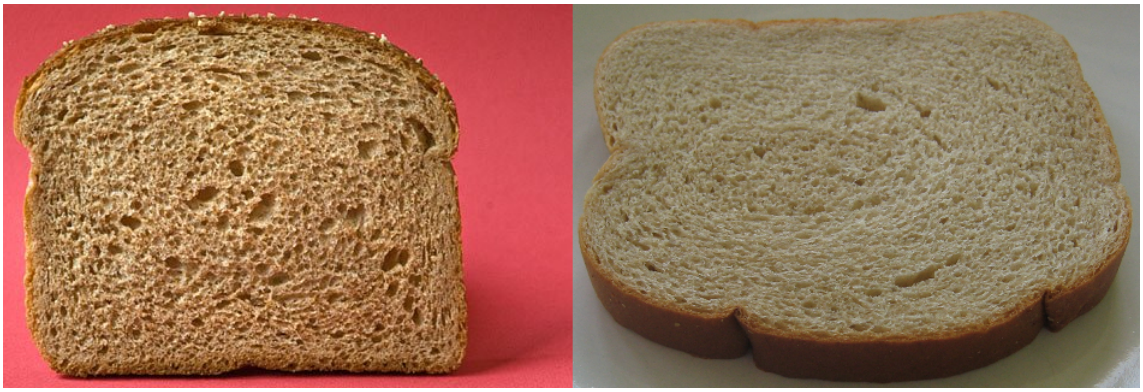

Figura 2: Imágenes de cortes reales de pan

Por otro lado, los sistemas dinámicos previamente presentados producen formas naturales (ver Fig. 3). En las imágenes pueden observarse círculos y espirales, entre otras formas. Las imágenes se obtuvieron dibujando trayectorias sobre un plano, siguiendo distintas ODEs. Tres ODEs describen las dinámicas presentes en las imágenes. A modo de ejemplo, la imagen de la izquierda es el resultado del siguiente conjunto de ecuaciones:

$$\begin{aligned} \dot{x} &= x^2 - y^2 + 1, \\ \dot{y} &= 2xy + 1. \end{aligned} \tag{4}$$

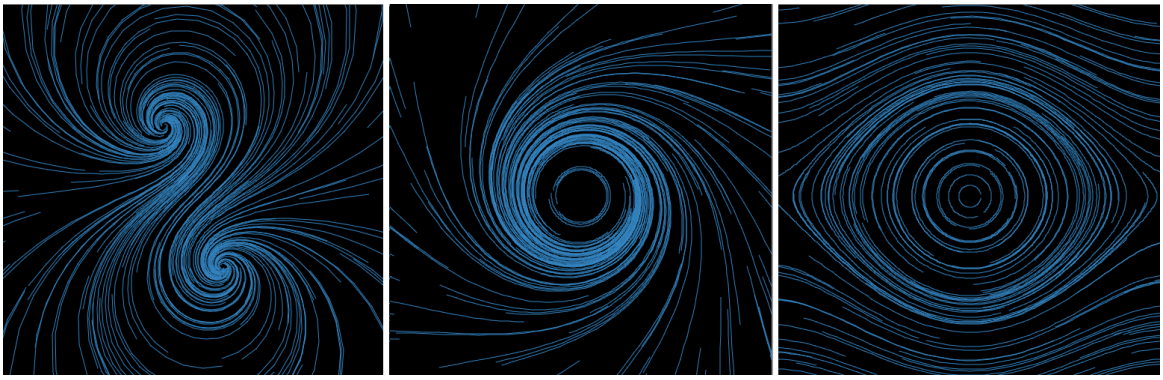

Figura 3: Sistemas dinámicos en el plano.

En los ejemplos se eligen posiciones aleatorias y luego el sistema se resuelve por medio de un *solver* Runge-Kutta de cuarto orden, el cual permite conocer la dirección a tomar en cada punto por la trayectoria (en las imágenes se computaron trayectorias hacia adelante y hacia atrás en el tiempo para lograr una mejor visualización). La imagen de la izquierda muestra un

atractor y un repeledor claramente visibles. El centro de la espiral más a la izquierda es un atractor (a medida que t avanza, las trayectorias convergen hacia el punto), mientras que el otro centro es un repeledor. Los atractores pueden no ser puntuales, como muestran las restantes dos imágenes. La imagen de la derecha muestra atractores en forma de círculos (las trayectorias ciclan por el círculo).

Las partículas producen distintos patrones al seguir las trayectorias definidas en el plano y el espacio. Esto se logra resolviendo numéricamente el sistema dinámico en la posición agregada a la partícula, seleccionando como siguiente posición de crecimiento aquella posición del contorno que mejor aproxima la solución del sistema dinámico (para esto, sólo se agrega al contorno esa posición). El siguiente pseudo-código muestra cómo se modifica el algoritmo de modelado para incluir este comportamiento:

Algoritmo 2 Modificación del algoritmo de modelado por medio de sistemas dinámicos

$L[h] \leftarrow 0$ ▷ masa ->aire
$O[i].agregar(h)$
$solucion \leftarrow Runge\_Kutta(h)$ ▷ Se calcula la siguiente posición
$vec = vecindario(h)$
$mejor = abs(vec[0] - solucion)$
$elegida = h$
$vec.eliminar(h)$
for $w \in vec$ **do**
    // Se calcula la posición del vecindario que mejor aproxima al sistema
    if $abs(vec[w] - solucion) < mejor$ **then**
        $mejor = abs(vec[w] - solucion)$
        $elegida = w$
    if $aleatorio() > 1 - aleatoriedad$ **then** ▷ $0 <= aleatorio() <= 1$
        $C[i].agregar(w)$
// Se agrega al vecindario sólo la posición que mejor aproxima la solución
$C[i].agregar(elegida)$

Las partículas se deforman de forma global en un patrón que es visualmente similar a las trayectorias que produce el sistema (ver Fig. 4). En las imágenes, de izquierda a derecha se decrementa la *aleatoriedad* de las trayectorias. El parámetro aleatoriedad seteado a $0{,}1$ genera la imagen de la derecha, lo cual significa que las burbujas eligen como siguiente posición de crecimiento la que mejor se acopla al sistema con una probabilidad de $0{,}9$. La probabilidad se define como $1 - aleatoriedad$, donde $0 \leq aleatoriedad \leq 1$. Las ecuaciones del sistema son las mismas que las de la imagen derecha mostrada en la Fig. 3. Estos patrones pueden ser utilizados además en otros materiales cocidos, variando el parámetro de aleatoriedad. Distintos sistemas de ecuaciones pueden utilizarse para definir distintos patrones.

La siguiente sección muestra cómo estas particulas modificadas son renderizadas.

2.4. Algoritmo de renderizado

En esta sección explicaremos la teoría de la técnica de DVR y se presentarán algunos aspectos de la implementación utilizada en este trabajo.

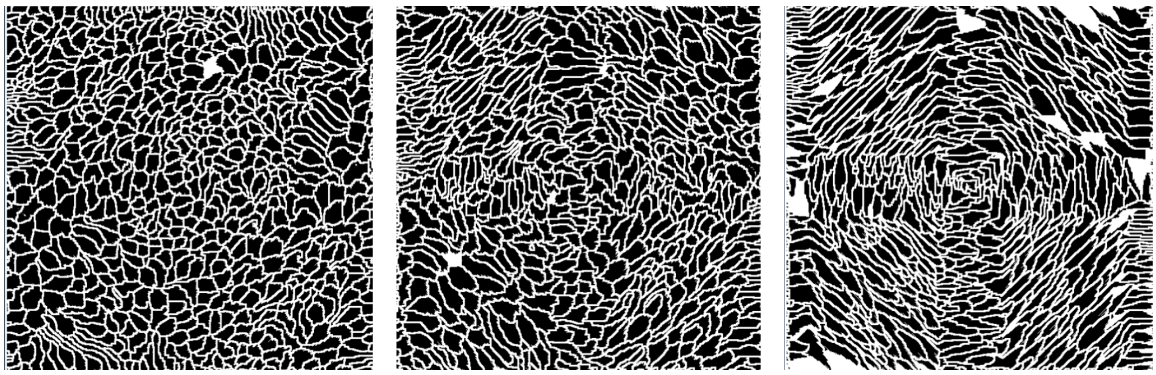

Figura 4: Sistemas dinámicos aplicados en sistemas de partículas. Efecto del paámetro *aleatoriedad*. De izquierda a derecha, aleatoriedad: 0.3,0.2,0.1 respectivamente.

2.4.1. *Direct Volume Rendering*

La técnica de DVR (renderizado directo de volúmenes en español) tiene como objetivo crear una representación bidimensional de un volumen definido por una función de densidad tridimensional. Para ello, se emiten rayos desde el punto de vista de una cámara en una escena virtual y se utiliza la función de densidad para calcular la cantidad de luz que la cámara recibe en la dirección del rayo. Para esto se evalúa la función de densidad en el camino del rayo y se usan los valores adquiridos para aproximar el efecto de varios fenómenos lumínicos, como pueden ser la extinción, transmitancia, o dispersión lumínica, entre otros. La información obtenida de procesar todos los rayos se utiliza para definir el color de los pixeles en la imagen final.

La radiancia es la cantidad de luz que pasa, o es emitida, desde un punto y atraviesa un determinado ángulo sólido. En el contexto de DVR, el medio que los rayos atraviesan, y que es definido por una función de densidad, es considerado como emisivo. Por lo tanto, cuando se busca calcular la cantidad de luz recibida en la dirección de un rayo, lo que se hace es aproximar la radiancia recibida de un punto distante siguiendo la dirección del rayo. El valor de la radiancia es aproximado como la suma de una radiancia de fondo y la radiancia emitida por el medio por el cuál se mueve el rayo (Kratz, 2006) :

$$L(p_n) = L_b + \int_{p_0}^{p_n} \frac{\partial L(t)}{\partial p}\, dt, \tag{5}$$

donde L_b es la radiancia de fondo, p_0 y p_n son los puntos inspeccionados en la dirección del rayo más cercano y más lejano respectivamente, $L(t)$ es la radiancia evaluada en el punto t, y ∂p es la distancia entre puntos evaluados. En el momento de calcular $L(p_n)$, la integral es aproximada por una suma.

La extinción es la pérdida de fotones en un haz de luz debido a la absorción en el medio que atraviesa y la dispersión hacia otras direcciones. Algunos de los fotones colisionarán con particulas del medio y serán absorbidas y transformadas en otras formas de energía, mayormente calor. Otras rebotarán y pasarán a moverse en otras direcciones. Estos fenómenos se aproximan usando un coeficiente de absorción para el medio, k_a y un coeficiente de dispersión k_s. Si el efecto de dispersión es ignorado, la fórmula que define la cantidad de radiancia absorbida en el largo de un segmento de rayo es:

$$L_b\, e^{-\int_{p_0}^{p_n} k_a(t)\, dt}. \tag{6}$$

El valor $\int_{p_i}^{p_j} k_a(t)\, dt$ es llamado coeficiente de absorción y se referenciará como $\tau_{(p_i,p_j)}$.

La transmitancia es un concepto complementario a la extinción y describe la cantidad de luz que pasa por un medio en una dirección determinada. El valor de transmitancia entre dos puntos p_i y p_j es:

$$T(p_i, p_j) = e^{-\tau_{(p_i,p_j)}}. \tag{7}$$

Si la emisión de luz se asume como un término constante (ρ) para todos los puntos del medio, la fórmula inicial de radiancia queda:

$$L(p_n) = L_b\, e^{-\tau(p_0,p_n)} + \int_{p_0}^{p_n} \rho\, e^{-\tau(t,p_n)}\, dt. \tag{8}$$

Esto significa que la radiancia entre los puntos p_0 y p_n se puede calcular como la radiancia de fondo restante luego de la atenuación del medio sumada a la emisión, también atenuada, en todos los puntos del medio que atraviesa el rayo.

La técnica de DVR define un volumen donde una función de densidad se evalúa en intervalos regulares y utiliza esa información para aproximar la transmitancia en esos puntos y de esa manera aproximar la cantidad de luz que llega a la cámara. La suma integral descrita anteriormente se reemplaza por una suma discreta de los puntos evaluados de un rayo donde éste intersecta al volumen que interesa representar.

Otros efectos lumínicos pueden ser aproximados. Esto aumenta la fidelidad de la imagen final pero también aumenta el costo de cómputo de la técnica. Algunos de estos efectos son el cálculo de fase, el cálculo de luz entrante por dispersión o luz extinguida por dispersión, entre otros. Dado que el objetivo de este trabajo es lograr un renderizado en tiempo real, el algoritmo implementado usa como base el modelo de cálculo de radiancia simplificado que toma en cuenta sólo la transmitancia del medio.

2.4.2. Renderizado de pan usando *direct volume rendering*

Se creó un programa de prueba[1] para evaluar el sistema de partículas que describe la estructura del pan. Este programa usa el sistema de partículas para generar una textura volumétrica que se interpreta como una función de densidad. Esta textura precomputada se usa como entrada para un motor gráfico que utiliza la técnica de DVR para generar imágenes del pan representado por el sistema de partículas original. Este programa demuestra que el método de renderizado propuesto es compatible con los motores gráficos basados en técnicas de renderización en tiempo real en GPU. Se obtienen imágenes de un material realístico así como efectos de sombras suaves dentro del volumen. Esto significa que las técnicas usadas para renderizar estos materiales pueden ser integradas en cualquier motor de renderizado que soporte shaders.

La malla que utiliza el modelo es un cubo que contiene el volumen definido por el sistema de partículas original. El código del shader de vértices es muy simple, proveyendo sólamente información geométrica al shader de fragmentos. Este último es donde se encuentra la mayor parte de los cálculos a realizar.

Dentro del shader de fragmentos la primera operación es calcular la geometría de un rayo cuyo origen es la cámara de la escena y cuya dirección lo lleva hacia el fragmento siendo calculado. Este rayo es recorrido en intervalos regulares, evaluando la textura volumétrica para obtener la densidad del pan en esos puntos. Este valor se utiliza para calcular la transmitancia

[1] disponible en *https://www.github.com/rbaravalle/Pysys*

acumulada desde la cámara hasta el punto evaluado. Una vez que la transmitancia es menor que un valor preestablecido o el rayo sale del cubo que define el volumen, el cómputo termina.

En cada punto evaluado también se computa la transmitancia dentro del volumen desde el punto hacia la fuente de luz en la escena. Esto se hace emitiendo un rayo desde el punto con dirección a la luz y nuevamente calculando la densidad en varios puntos del rayo. Con esta nueva información se aproxima la cantidad de luz que llega directamente al punto considerado, y permite representar sombras dentro del volumen.

La información de transmitancia de los puntos evaluados del rayo principal y desde estos puntos hacia la luz se utilizan para calcular el color final del fragmento. A partir de esta información y tomando diferentes consideraciones artísticas pueden lograrse representaciones realísticas de diferentes materiales. En el caso de las imágenes de muestra presentadas en este trabajo el color del fragmento será más oscuro para areas del volumen que se consideran dentro de la corteza del pan y será de un color amarillo claro para la miga. También se usa un componente especular tenue. La información de transmitancia entre los puntos evaluados y la luz ayuda a proveer detalles de la estructura del pan. La Fig 5 muestra un esquema del cálculo del color final del pixel.

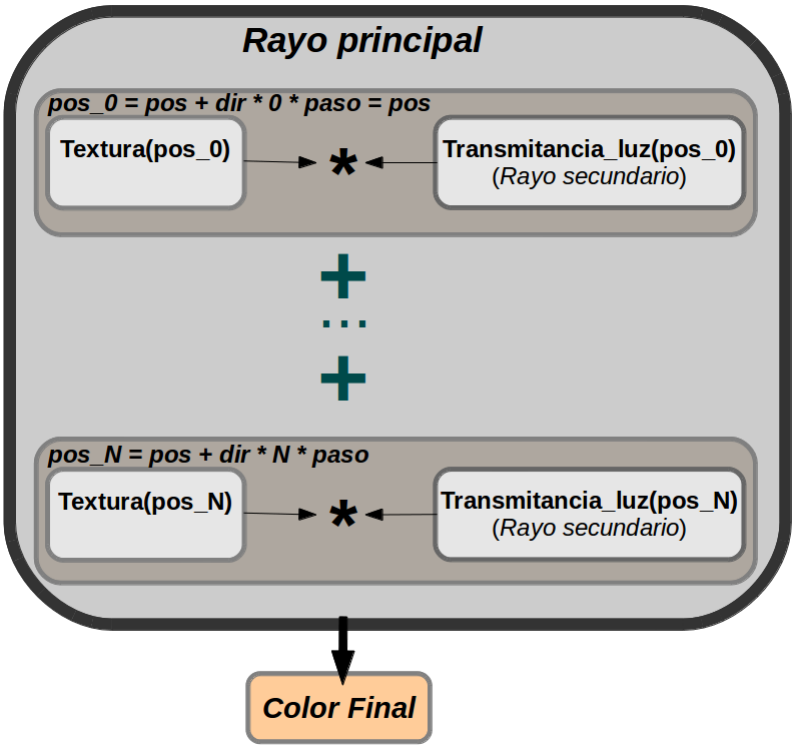

Figura 5: Cálculo de color en el shader de fragmentos.

El programa de prueba creado permite modificar parámetros tales como el coeficiente de transmitancia del pan, el límite de transmitancia, el color asignado a la miga y la intensidad de los reflejos especulares, entre otros. Esta capacidad permite crear imágenes que semejan otros materiales porosos, como por ejemplo esponjas.

Corteza, fetas y cortes Las partes del volumen pertenecientes a la corteza y a la miga del pan fueron determinadas manualmente mediante una función que asigna una u otra propiedad a cada punto del volumen a partir de su posición. Por ejemplo, para volúmenes de forma cilíndrica se

definió una función que designa como corteza a los puntos que están a más de una distancia predefinida del eje del cilindro.

De la misma manera se define una función que determina si puntos del volumen deben considerarse vacíos, independientemente del valor de la textura volumétrica en ese punto. Esto permite definir fetas de pan de manera sencilla. Un ejemplo del uso de este mecanismo puede apreciarse en la Fig. 6.

La asignación de vacío y de corteza deben ser extendidos más allá de ecuaciones basadas en posiciones para poder ser usadas en un proceso artístico. En la siguiente sección se presentan y se evalúan los resultados obtenidos.

3. RESULTADOS

En esta sección se detallan las imágenes y los tiempos de renderizado obtenidos.

3.1. Resultados del renderizado

Las imágenes obtenidas a partir del método descrito en la sección anterior fueron renderizadas en una computadora con una placa gráfica nVidia GTX 480 (480 cores), la cual es normalmente de uso hogareño. La CPU fue una Intel(R) Core(TM) i5-2300 CPU (cuatro procesadores). La resolución de las imágenes es de 1440×990 pixels. Se obtuvieron diferentes imágenes que semejan materiales horneados. Diferentes tipos de pan pueden ser representados variando los parámetros de transmitancia y colores utilizados (ver Fig. 6). En la imagen central los patrones producidos por los sistemas de partículas descritos en las secciones previas son claramente visibles. En ese caso, el tiempo de vida de las partículas es diferente para cada una y de esa manera se obtienen burbujas de diferentes tamaños.

Figura 6: Imágenes de diferentes tipos de pan renderizados en tiempo real. La imagen de la derecha muestra un pan sin corteza

Es posible obtener otros materiales (ver Fig. 7). Estos son el resultado de la variación de parámetros técnicos y artísticos del modelo. En las imágenes de prueba pueden distinguirse un budín (izquierda), un pedazo de torta (medio) y una esponja (derecha). En el caso de la esponja se modificaron los parámetros que definen la función de densidad. Cuando no hay levadura en el proceso de creación puede utilizarse una textura volumétrica cuyos valores provienen de una función aleatoria. La retroiluminación es también aproximada con este modelo (ver Fig.8). En esa imagen puede apreciarse una esponja retroiluminada junto con la propagación de luz a través del volumen que representa.

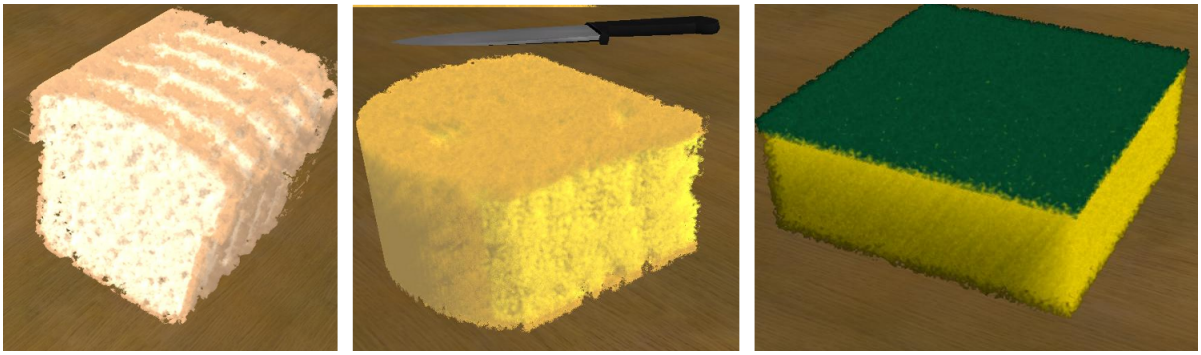

Figura 7: Distintos materiales obtenidos a partir de diferentes configuraciones de parámetros. De izquierda a derecha: budín, torta y esponja.

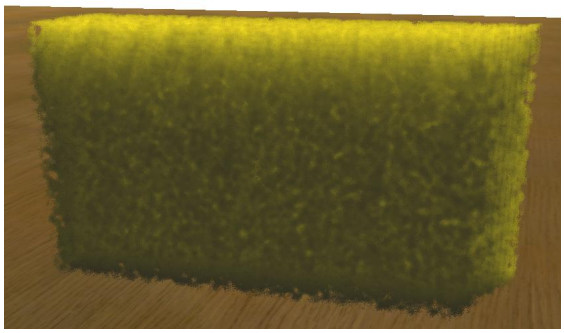

Figura 8: Esponja retroiluminada.

3.2. Tiempos de renderizado

La mayoría de las imágenes se obtuvieron con tasas de refresco de tiempo real (más de 30 FPS), como muestra la Tabla 1. La eficiencia del proceso se resiente cuando la transmitancia es muy baja (el material es casi transparente), dado que se evaluarán más puntos en los rayos a recorrer antes de llegar al límite de transmitancia. Otro parámetro importante es la distancia entre puntos a evaluar. En la Tabla 2 se observa que a medida que la cantidad de rayos y de pasos del rayo aumenta, la velocidad decrece. La tabla muestra que los rayos secundarios constituyen el principal cuello de botella, lo cual es lógico, dado que a cada paso del rayo principal, el mismo computa un rayo secundario hacia la luz.

Experimentalmente se encontró que para todos los casos evaluar 100 puntos o más presenta buenos resultados.El proceso escala automáticamente con el número de procesadores en una GPU, por lo cual la tasa de refresco obtenida será mayor en GPUs más rápidas y de más procesadores.

	Pan 1	Pan 2	Pan 3	Budín	Torta	Esponja
FPS promedio	32.2	75.5	45.2	28.5	54.2	29.7
Puntos de evaluación	140	140	140	256	140	256
Transmitancia	15	15	15	15	15	2.25

Tabla 1: Tiempos de renderizado y parámetros de las imágenes de prueba.

Pasos del rayo	128	256
Tiempo total shaders	10 ms	32.5 ms
Rayo Principal	2 ms	5 ms
Rayos Secundarios	8 ms	27.5 ms

Tabla 2: Detalle de tiempos de renderizado en milisegundos.

3.3. Discusión

Según el conocimiento de los autores, éste es el primer intento de llevar a cabo una renderización en tiempo real de pan de manera convincente sin el uso de procesos intermedios complicados (captura de imágenes, generación de mallas, post-procesamiento). Existen buenos resultados de renderizado de pan obtenidos con otros métodos (Cho et al., 2007), pero es difícil comparar ese trabajo con el presentado en este artículo debido a que ni los detalles de la técnica utilizada ni los tiempos de cálculo han sido publicados.

Dentro del volumen a renderizar se pueden definir regiones con diferentes propiedades. Esta idea permite generar imágenes con miga y corteza con diferentes parámetros.

La integración de la técnica descrita con motores gráficos es simple. La información de profundidad de los fragmentos puede obtenerse de manera sencilla y por lo tanto pueden utilizarse técnicas populares de sombras, tales como mapas de sombras.

Los tiempos de cómputo muestran una alta eficiencia del proceso, lo cual depende en gran medida del número de puntos de evaluación usados y la transmitancia del material. Se pueden alcanzcar tiempos de cómputo consistentes con aplicaciones de tiempo real en todos los casos menos en los cuales el volumen ocupa la mayor parte de la imagen a generar, dado que la técnica se calcula casi enteramente en los shader de fragmentos.

Los resultados obtenidos pueden extenderse de diferentes formas, las cuales se mencionan en la siguiente sección.

4. CONCLUSIONES

En este trabajo se aplica el modelo de transmitancia en DVR a un campo escalar 3D el cual representa la estructura de la miga de pan, con el objetivo de obtener imágenes foto-realísticas del material. Esta estructura se genera utilizando sistemas de partículas, los cuales evolucionan siguiendo sistemas dinámicos de una manera probabilística. La simulación se resuelve numéricamente. Los resultados muestran imágenes con gran resemblanza a imágenes del material en tiempo real. Los métodos propuestos pueden aplicarse en distintas áreas, como juegos serios (Susi et al., 2007) y rendering foto-realístico. El método de renderizado no presenta las desventajas de otros métodos del estado del arte.

La principal desventaja del método (la cual está presente también en los demás métodos del estado del arte) es la resolución, ya que las placas gráficas tienen un límite en las dimensiones de texturas que soportan. Para resolver este problema, se propone como trabajo futuro setear diferentes texturas volumétricas dependiendo de la distancia de la cámara al volumen.

Otras posibles continuaciones a este trabajo incluyen mejorar el algoritmo de renderizado, buscando tomar en cuenta iluminación indirecta y difusión de la luz dentro del material. Además, se tratarán ecuaciones diferenciales más generales, para hacer posible una aplicación directa del modelo matemático del proceso de cocción. Se investigarán otros materiales como quesos. Otra idea interesante sería definir primitivas de miga y corteza permitiendo un modelado artístico de estas características.

REFERENCIAS

Baravalle R., Delrieux C., y Gómez J.C. Síntesis de texturas utilizando sistemas de partículas. ECImag 2011. Actas de la Tercera Escuela y Workshop Argentino en Ciencias de las Imágenes. 2011.

Cho J.H., Xenakis A., Gronsky S., y Shah A. Anyone can cook - inside ratatouille's kitchen. En *ACM SIGGRAPH 2007 Courses*. ACM, New York, NY, USA, 2007.

Cords H. Moving with the flow: Wave particles in flowing liquids. *Journal of WSCG*, 16(1-3):145–152, 2008.

Harris M. y Luebke K. GPGPU Tutorial. Supercomputing 2006 Conference, 2006.

Kenny Yeo. Voodoo Beginnings - 10 Years of GPU Development. *PcStats*, 2009.

Kratz A. *Advanced Illumination Techniques for GPU-Based Direct Volume Rendering*. Tesis de Maestría, Koblenz - Landau, Wien, Germany, 2006.

Kurt M. y Edwards D. A survey of brdf models for computer graphics. *SIGGRAPH Comput. Graph.*, 43(2):4:1–4:7, 2009.

Levoy M. Display of surfaces from volume data. *Computer Graphics and Applications, IEEE*, 8(3):29–37, 1988.

Lorensen W.E. y Cline H.E. Marching cubes: A high resolution 3d surface construction algorithm. En *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, páginas 163–169. ACM, New York, NY, USA, 1987.

Mandelbrot B.B. *The fractal geometry of nature*. W.H. Freeman, 1 edición, 1982.

Mondal A. y Datta A. Bread baking - a review. *Journal of Food Engineering*, 86(4):465 – 474, 2008.

Müller M., Charypar D., y Gross M. Particle-based fluid simulation for interactive applications. En *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, páginas 154–159. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2003.

Paiva A., Petronetto F., Lewiner T., y Tavares G. Particle-based viscoplastic fluid/solid simulation. *Computer-Aided Design*, 41(4):306 – 314, 2009.

Perlin K. y Hoffert E.M. Hypertexture. *SIGGRAPH Comput. Graph.*, 23(3):253–262, 1989.

Reeves W.T. Particle systems – a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.*, 2:91–108, 1983.

Scanlon M. y Zghal M. Bread properties and crumb structure. *Food Research International*, 34(10):841 – 864, 2001.

Shah P., Campbell G., Mckee S., y Rielly C. Proving of bread dough: Modelling the growth of individual bubbles. *Food and Bioproducts Processing*, 76(2):73 – 79, 1998.

Stam J. Stable fluids. En *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, páginas 121–128. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999.

Strogatz S.H. *Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry, And Engineering (Studies in Nonlinearity)*. Studies in nonlinearity. Westview Press, 1 edición, 2001.

Susi T., Johannesson M., y Backlund P. Serious games: An overview. Informe Técnico, Institutionen för kommunikation och information, University of Skövde, Sweden, 2007.

Tong X., Wang J., Lin S., Guo B., y Shum H.Y. Modeling and rendering of quasi-homogeneous materials. *ACM Trans. Graph.*, 24(3):1054–1061, 2005.

Voglsam G. *Real-time Ray Tracing on the GPU - Ray Tracing using CUDA and kD-Trees*. Tesis

de Maestría, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 2013.