Universidad del Valle de Guatemala

Facultad de ingeniería

Computación Paralela y Distribuida

Catedrático: Luis García

Proyecto 1

Nelson Eduardo García Bravatti 22434
Gabriel Paz 221087
Joaquín Puente 22296

Guatemala, 5 de septiembre de 2025

Introducción:

Se presenta el desarrollo de un screensaver de ‘lluvia’ que simula las ondas radiales producidas por el impacto de gotas sobre una superficie de agua. El modelo implementa un campo de alturas H(x,y,t) generado por la superposición lineal de contribuciones individuales de cada gota. La versión actual es secuencial y está estructurada para habilitar una paralelización posterior con OpenMP. El usuario especifica únicamente el tamaño de la ventana y el número de gotas N; el sistema muestra los FPS para asegurar una experiencia fluida (≥ 30 FPS).

Antecedentes:

OpenMP es una API estándar para programación paralela en memoria compartida en C/C++ y Fortran, ampliamente soportada en sistemas Windows, macOS y Linux. En el ámbito físico, el principio de superposición establece que, cuando varias ondas coinciden, la perturbación total es la suma de sus perturbaciones individuales, produciendo interferencia constructiva o destructiva según su fase relativa. Estos conceptos fundamentan la versión paralela (por píxel o por ‘tiles’) del algoritmo.

Arquitectura General:

La implementación separa claramente la simulación (cálculo del campo H) del renderizado (SDL2 en el hilo principal):

• Captura y validación de argumentos (--width, --height, --N).

• Inicialización de SDL2: ventana, renderer y textura ARGB8888.

• Mundo (World): gestión de gotas activas, parámetros aleatorios y respawn por vida útil.

• Simulación: acumulación de H(x,y) mediante la suma de contribuciones ‘ripple_contrib’ por gota.

• Sombreado: cálculo de normales por diferencias finitas y composición fotométrica (Lambert + Blinn, Fresnel, ambiente de cielo y absorción espectral) para un aspecto acuático realista.

• Presentación: copia del buffer a la textura SDL y presentación en pantalla, con FPS en el título.

Modelo Físico y Superposición:

Cada gota d con tiempo de impacto t0 contribuye con un anillo principal centrado en $r \approx c\cdot(t-t0)$, con envolvente gaussiana de ancho σ, amortiguamiento temporal e atenuación radial suave. Para

mayor realismo se incluyen ripples capilares (gaussianas estrechas adyacentes a la cresta) y una salpicadura inicial breve con modulación angular. El height field H(x,y,t) resulta de sumar las contribuciones de todas las gotas. La interferencia emerge de forma natural al superponer crestas y valles, reforzando o atenuando la amplitud en distintas regiones.

Paralelización Prevista con OpenMp:

La función de acumulación del campo H se presta a paralelización con OpenMP. Existen dos estrategias principales:

1) Paralelización por píxel: distribuir el doble bucle (y,x) y sumar las N gotas por píxel (sencillo, buen balance; costo computacional estable).

2) Paralelización por tiles/ROI: para cada gota, procesar únicamente la banda activa alrededor del anillo (culling), usando acumuladores por hilo y combinación final para evitar condiciones de carrera.

En ambos casos, SDL debe permanecer en el hilo principal; únicamente la simulación se paraleliza.

Entradas, Salidas y Parámetros:

Entradas: tamaño de ventana (--width, --height) y N gotas (--N). Opcionalmente se pueden exponer semilla, paletas y parámetros estéticos/numéricos en futuras versiones.

Salidas: ventana con render en tiempo real y título con FPS suavizados (~1 s).

Conclusiones y Recomendaciones:

La versión secuencial logra un aspecto de agua verosímil combinando un modelo de ondas simple pero efectivo y un sombreado basado en normales. Para cargas mayores (resoluciones altas o N grande), se recomienda paralelizar la acumulación del campo H con OpenMP, ya sea por píxel o por tiles con culling por anillo. Como trabajo futuro: exponer parámetros estéticos por CLI, añadir modo headless para perfilado, y evaluar técnicas SIMD y caches amistosas para maximizar el speedup y la eficiencia.

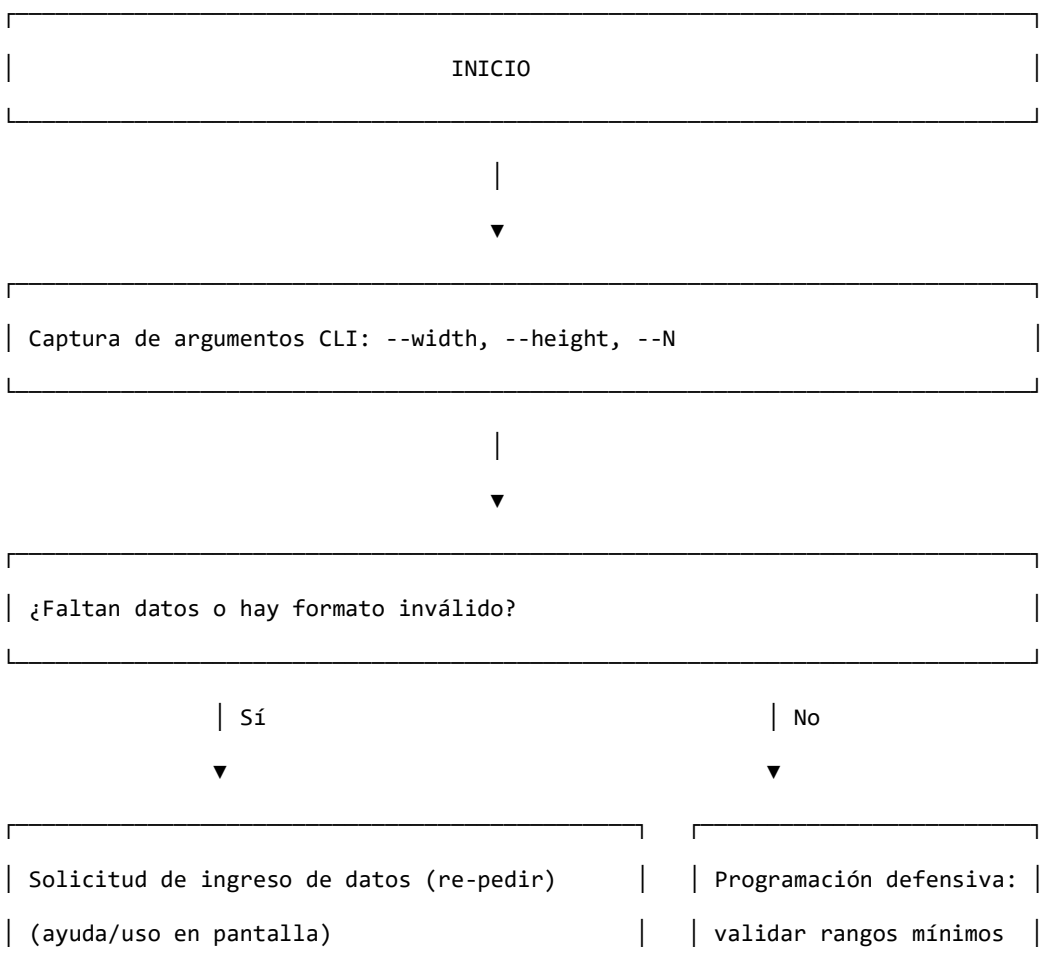
Bibliografía:

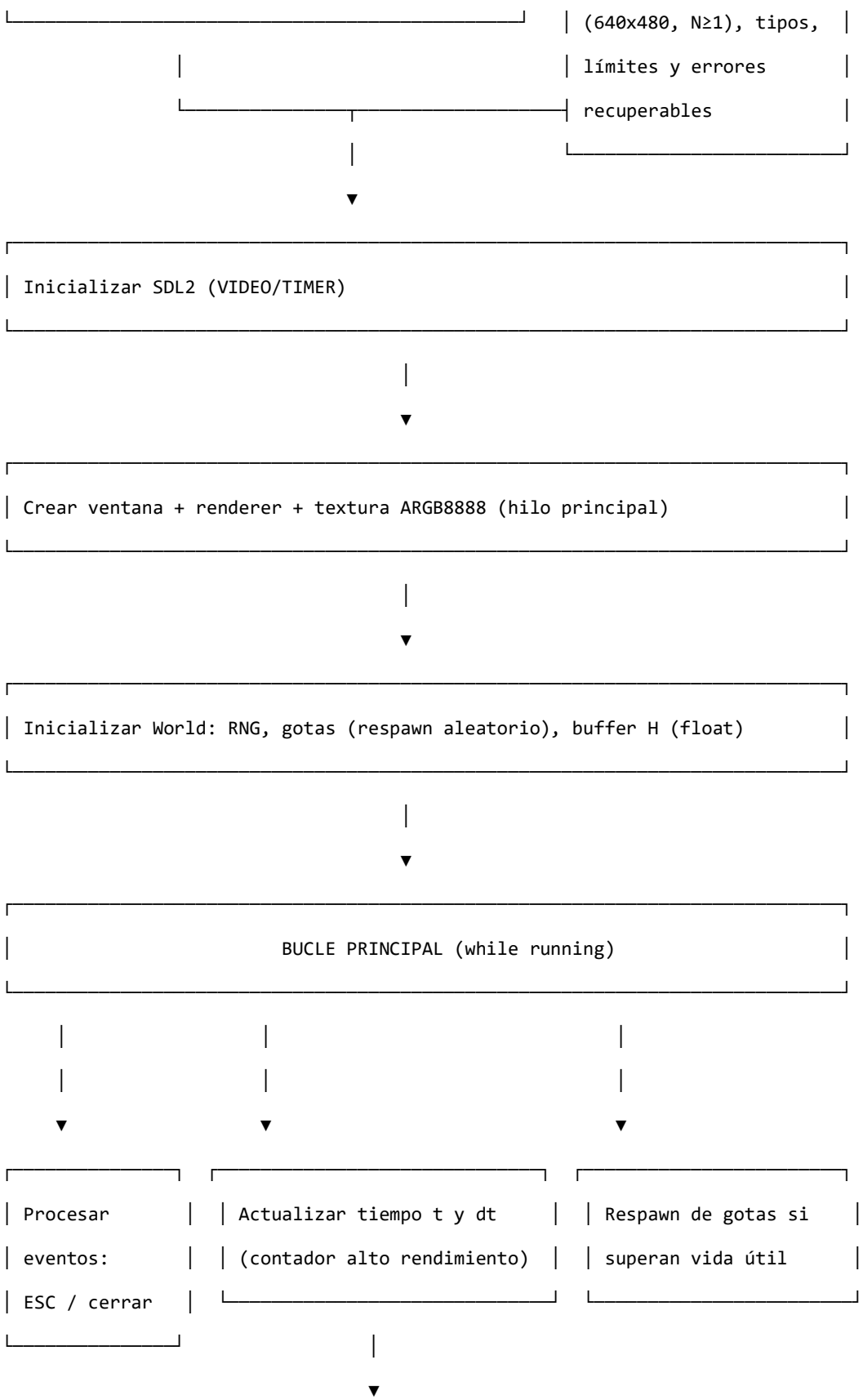
Urone, P. P., & Hinrichs, R. (2020, March 26). 13.3 Wave Interaction: Superposition and Interference - Physics | OpenStax. https://openstax.org/books/physics/pages/13-3-wave-interaction-superposition-and-interference

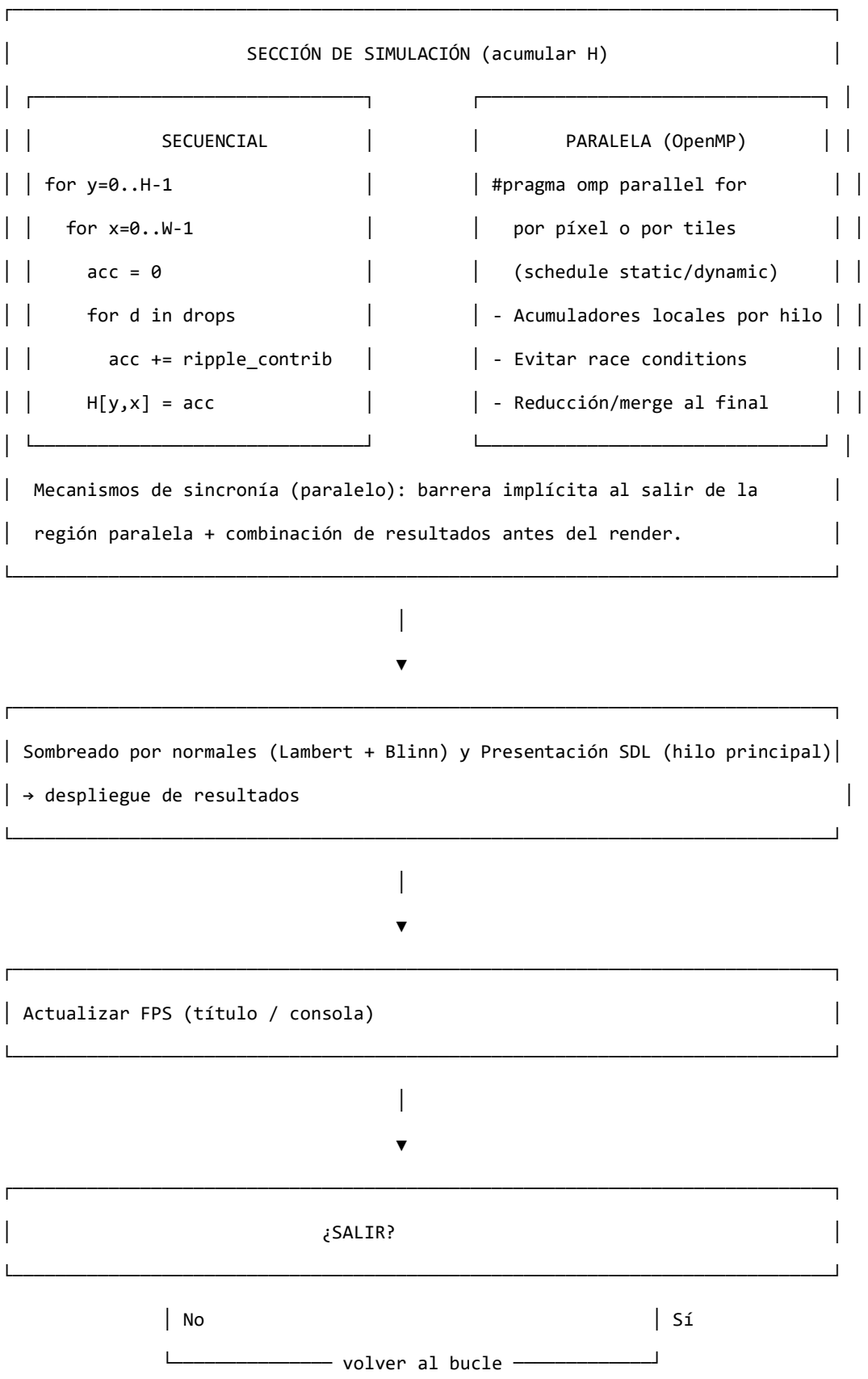
Tim.Lewis. (2025, June 27). Specifications - OpenMP. OpenMP. https://www.openmp.org/specifications/

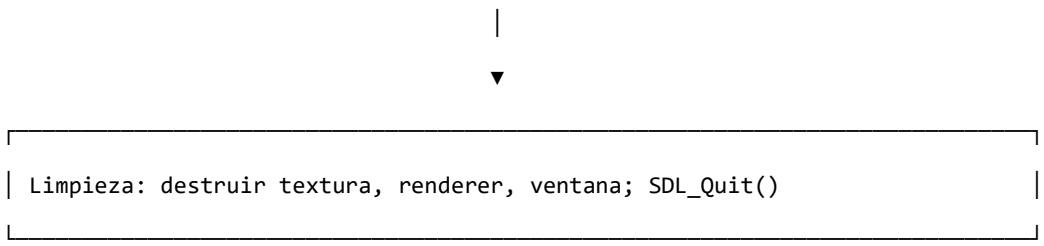
SDL2/FrontPage. (n.d.). SDL2 Wiki. https://wiki.libsdl.org/SDL2/FrontPage

Anexo 1: Diagrama de Flujo del programa




(640x480, N≥1), tipos,
límites y errores
recuperables
Inicializar SDL2 (VIDEO/TIMER)
Crear ventana + renderer + textura ARGB8888 (hilo principal)
Inicializar World: RNG, gotas (respawn aleatorio), buffer H (float)
BUCLE PRINCIPAL (while running)
Procesar eventos: ESC / cerrar
Actualizar tiempo t y dt (contador alto rendimiento)
Respawn de gotas si superan vida útil


SECCIÓN DE SIMULACIÓN (acumular H)
SECUENCIAL
for y=0..H-1
for x=0..W-1
acc = 0
for d in drops
acc += ripple_contrib
H[y,x] = acc
PARALELA (OpenMP)
#pragma omp parallel for
por píxel o por tiles
(schedule static/dynamic)
- Acumuladores locales por hilo
- Evitar race conditions
- Reducción/merge al final
Mecanismos de sincronía (paralelo): barrera implícita al salir de la región paralela + combinación de resultados antes del render.
Sombreado por normales (Lambert + Blinn) y Presentación SDL (hilo principal)
→ despliegue de resultados
Actualizar FPS (título / consola)
¿SALIR?
No
Sí
volver al bucle



Anexo 2: Catálogo de Funciones

Función / Clase	Entradas (tipo)	Salidas (tipo)	Descripción
int main(int,char**)	--width:int, --height:int, --N:int	int (código de salida)	Parseo y validación de argumentos, init SDL, loop principal, FPS y limpieza.
World (struct/clase)	AppConfig, RNG interno	Gestor de estado (drops, H)	Administra gotas activas, respawn aleatorio según vida útil y almacena el buffer H.
void respawn_drop(Drop&, float now)	Drop&, now:float	—	Reinicializa parámetros de una gota (posición, fuerza, vida, capilares, splash).
void maybe_respawn(float now)	now:float	—	Respawnea gotas cuya vida superó el umbral para mantener actividad visual continua.
float ripple_contrib(float x,float y,float t,const Drop& d)	x,y,t, d	float	Contribución de una gota en (x,y,t): cresta principal + capilares + splash con amortiguamiento.
void accumulate_heightfield_sequential(std::vec	H, W, Hh, drops, t	H (modificado)	Acumula el campo de alturas sumando las contribuciones de

tor<float>& H, int W, int Hh, const std::vector<Drop>& drops, float t)			todas las gotas (superposición lineal).
void shade_and_present(SDL_Renderer*, PixelBuffer&, const std::vector<float>& H, float slopeScale)	renderer*, textura, H, slopeScale	Frame en pantalla	Calcula normales por diferencias finitas y compone color (Lambert+Blinn, Fresnel, absorción).

Anexo 3: Bitácora de pruebas