# Informe Proyecto 01 - Screensaver Paralelo (Lissajous Particles)

1. Resumen

Se desarrolló un screensaver en C++/SDL2 cuyo efecto visual está basado en partículas que exhiben una trayectoria de curvas de Lissajous. Se implementaron dos versiones de la simulación: secuencial y paralela (OpenMP), con un modo de benchmark que mide updates per second (UPS). A partir de múltiples corridas con distintos tamaños de problema (N) y número de hilos, se calcularon speedup y eficiencia. En general, se observó un speedup cercano a 3 con 4–8 hilos para N medianos/grandes, y menor beneficio en N pequeños por sobrecostos de paralelización.

1. Diseño del algoritmo (Lissajous Particles)

Cada partícula evoluciona en el plano según ecuaciones paramétricas de tipo *Lissajous*:

Estas curvas son clásicamente conocidas como *figuras de Lissajous*; su forma depende de las frecuencias y fases relativas, y son habituales en osciloscopios.

Notas de implementación

* Se inicializan parámetros por partícula (centro, amplitudes, frecuencias, fases) de forma pseudoaleatoria (semilla reproducible).
* La actualización por partícula es independiente, lo que produce una complejidad O(N) y un patrón de paralelización sencillo.

1. Implementación

* Lenguaje/Librerías: C++17, SDL2 para ventana/renderizado; OpenMP para paralelizar el bucle de actualización.
* Render: círculos pequeños por partícula con un efecto blend para producir “estelas”.
* Paralelización: bucle for sobre el vector de partículas con #pragma omp parallel for.
* Argumentos en CLI: parámetros por línea de comandos (-n, --mode, --threads, --benchmark, --seed, etc.).

1. Metodología de medición

* Modo benchmark: corre la simulación sin ventana durante S segundos y reporta líneas [BENCH] con UPS (updates/segundo).
* Métrica: Para calcular el speedup se utilizó el promedio de UPS, no el “tiempo transcurrido” de la siguiente forma: ​
* Automatización: script bench.ps1 que compila y ejecuta una matriz de casos; resultados en CSV.

1. Resultados

Se procesaron los datos y se generaron tablas con promedios (10 repeticiones por caso), más un gráfico “Speedup vs Threads”:

A graph with lines and numbers

AI-generated content may be incorrect.

* 1. Hallazgos principales
* Mejor speedup por N (valor máximo encontrado):

|  |  |  |  |
| --- | --- | --- | --- |
| N | Speedup | # de Threads | Eficiencia |
| 1000 | 2.80 | 4 | 0.70 |
| 5000 | 2.91 | 8 | 0.36 |
| 10000 | 2.99 | 4 | 0.75 |
| 20000 | 2.96 | 4 | 0.74 |

* Tendencia: para valores pequeños de N el overhead de paralelizar pesa más y la eficiencia cae con muchos hilos; para N mayor, el speedup se estabiliza cerca de 3 con 4–8 hilos.
* El valor de eficiencia decrece con el número de hilos, como es esperable por porciones secuenciales y sobrecostos (planificación, cache, etc.), en línea con la ley de Amdahl.

1. Referencias

Wikipedia contributors. (2024). Lissajous curve. In Wikipedia. Recuperado de https://en.wikipedia.org/wiki/Lissajous\_curve

Wolfram Research. (s. f.). Lissajous Curve. In MathWorld—A Wolfram Web Resource. Recuperado de https://mathworld.wolfram.com/LissajousCurve.html

OpenMP Architecture Review Board. (2024). OpenMP Application Programming Interface, Version 5.2. Recuperado de https://www.openmp.org/wp-content/uploads/OpenMPRefGuide-5.2-Web-2024.pdf

Wikipedia contributors. (2024). Amdahl’s law. In Wikipedia. Recuperado de https://en.wikipedia.org/wiki/Amdahl%27s\_law

Bailey, M. J. (s. f.). Speedups and Amdahl’s Law [Apuntes de clase]. Oregon State University. Recuperado de https://web.engr.oregonstate.edu/~mjb/cs575/Handouts/speedups.and.amdahls.law.1pp.pdf

SDL. (s. f.). SDL\_RenderDrawPoint [Documentación de SDL2]. Recuperado de https://wiki.libsdl.org/SDL2/SDL\_RenderDrawPoint