Screensaver

Screensaver

Computación Paralela

Javier Alejandro Azurdia Arrecis | 21242

Angel Sebastian Castellanos Pineda | 21700

Diego Alejandro Morales Escobar | 21146

Introducción

Proyecto 1 de Computación Paralela

Este documento presenta el Proyecto 1 de Computación Paralela, desarrollado por los estudiantes Javier Alejandro Azurdia Arrecis, Angel Sebastian Castellanos Pineda, y Diego Alejandro Morales Escobar en la Universidad del Valle de Guatemala en agosto de 2024. El proyecto consiste en la implementación de un screensaver utilizando técnicas de programación paralela con OpenMP y la biblioteca SDL para la renderización gráfica.

El screensaver incluye tres componentes principales: líneas móviles, círculos y cuadrados/rectángulos. Las líneas se mueven suavemente por la pantalla formando patrones geométricos, los círculos interactúan entre sí y con las líneas, y los cuadrados/rectángulos también interactúan con los otros elementos. El programa permite configurar el número inicial de líneas y círculos, y muestra el rendimiento en términos de FPS (cuadros por segundo) durante la ejecución.

El proyecto se ha desarrollado en lenguaje C, utilizando la biblioteca SDL para la renderización gráfica y OpenMP para la paralelización de tareas. Se han implementado versiones secuenciales y paralelas del screensaver, con el objetivo de comparar su rendimiento y comportamiento bajo diferentes condiciones de carga. Se han realizado pruebas para evaluar el rendimiento de ambas versiones y se han analizado los resultados obtenidos.

Antecedentes

OpenMP

OpenMP (Open Multi-Processing) es una API diseñada para programación paralela en arquitecturas de memoria compartida. Fue introducida en 1997 como un estándar para aprovechar los procesadores multinúcleo, lo cual permite la ejecución de programas en paralelo para mejorar su rendimiento. OpenMP utiliza directivas de compilador, funciones de biblioteca y variables de entorno para manejar tareas como la paralelización de bucles y secciones de código, sincronización de hilos, y asignación de tareas.

Características clave:

- **Directivas sencillas:** Facilita la paralelización de programas C, C++ y Fortran mediante directivas que se agregan al código fuente.
- **Modelo de memoria compartida:** Todos los hilos tienen acceso a la misma memoria, lo cual simplifica la comunicación y el manejo de datos.
- **Control de concurrencia:** Ofrece mecanismos para sincronizar la ejecución de hilos, manejar secciones críticas, y reducir el riesgo de condiciones de carrera.
- **Escalabilidad:** Escala fácilmente en arquitecturas de múltiples núcleos, desde sistemas de escritorio hasta supercomputadoras.

(University of Notre Dame, s. f.)

SDL

SDL (Simple DirectMedia Layer) es una biblioteca de software libre escrita en C que proporciona una API de bajo nivel para acceso a hardware gráfico y multimedia, como gráficos 2D, audio, manejo de eventos, y entrada de dispositivos. Fue creada por Sam Lantinga en 1998 y ha sido ampliamente utilizada en el desarrollo de videojuegos, simuladores y otras aplicaciones multimedia.

Características clave:

- **Multiplataforma:** Funciona en una amplia variedad de sistemas operativos, incluyendo Windows, macOS, Linux, iOS, y Android.
- **Rendimiento:** Proporciona acceso directo a gráficos y audio para maximizar el rendimiento en tiempo real.
- **Simplicidad:** Ofrece una API sencilla que facilita el desarrollo de aplicaciones multimedia sin tener que lidiar con las complejidades del hardware subyacente.
- **Comunidad activa:** Cuenta con una gran comunidad y muchos recursos, lo que facilita el soporte y la colaboración en proyectos.

(SDL, 2023)

C

C es un lenguaje de programación de propósito general desarrollado en los años 70 por Dennis Ritchie en los laboratorios Bell. Es conocido por su eficiencia, control sobre el hardware y su influencia en muchos otros lenguajes modernos, como C++, C#, y Java.

Características clave:

- **Eficiencia y rendimiento:** Ofrece un control preciso sobre la memoria y la CPU, lo que lo hace ideal para el desarrollo de sistemas operativos, drivers y software de alto rendimiento.
- **Portabilidad:** Los programas en C pueden ser compilados y ejecutados en una amplia variedad de plataformas con cambios mínimos.
- **Lenguaje estructurado:** Permite la descomposición de problemas complejos en funciones, lo cual facilita la lectura y mantenimiento del código.
- **Amplio uso:** Sigue siendo uno de los lenguajes más utilizados en el desarrollo de software de sistemas, videojuegos, y aplicaciones embebidas.

(Ritchie, 2003)

Cuerpo

Ejecución del proyecto

Para poder ejecutar este proyecto primero se debe obtener el código fuente localizado en el repositorio

Una vez hecho esto se recomienda el uso del IDE CLion para la ejecución del mismo, en donde se debe abrir el proyecto llamadado `screensaver_p1` y ejecutar el archivo llamado `CMakeLists.txt`. Una vez hecho esto es posible ejecutar el archivo principal llamado `main.c`, el cual iniciará la ejecución del screensaver.

Instalación de SDL en CLion

El proyecto además utiliza la librería de SDL para poder renderizar en pantalla las figuras, es por esto que es necesario que el equipo cuente con dicha libería instalada. Para su correcta instalación se deben seguir estos pasos:

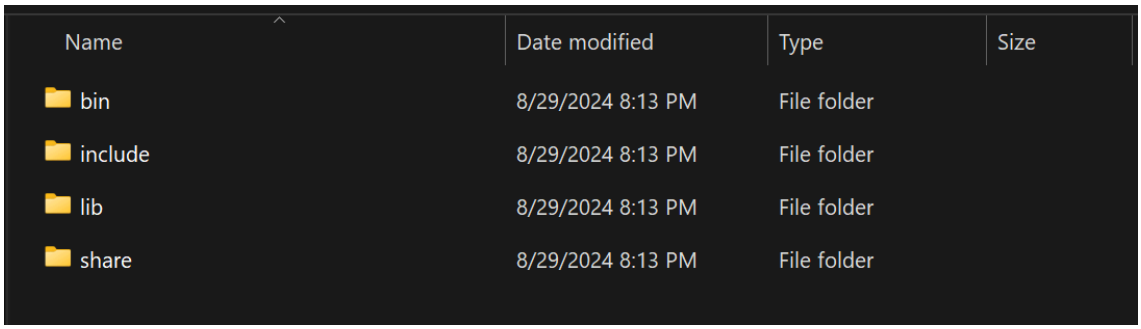
SDL este enlace contiene el archivo a descargar.

> **TIP**
>
> Este archivo es funcional para máquinas con el sistema operativo Windows con una arquitectura de 64-bits, en el siguiente enlace se pueden encontrar otros instaladores y el código fuente del mismo SDL Repository

Una vez hecho esto se debe descomprimir el `zip`

> **TIP**
>
> Debido a que la ruta de este archivo se usará más adelante se recomienda guardarlo en una ruta con fácil acceso y recordar la misma

Se debe navegar a la siguiente ruta dentro de los archivos descargados `.../SDL2-devel-2.30.6-mingw/SDL2-2.30.6/x86_64-w64-mingw32` y copiar la misma. Así mismo en esta ruta deberá encontrar archivos similares a los siguientes



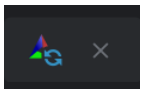
Name	Date modified	Type	Size
bin	8/29/2024 8:13 PM	File folder	
include	8/29/2024 8:13 PM	File folder	
lib	8/29/2024 8:13 PM	File folder	
share	8/29/2024 8:13 PM	File folder	

Con la ruta ya localizada, se debe dirigir al archivo llamado `CMakeLists.txt` y sustituirla, dejando un archivo similar al siguiente.

CMakeList.txt

```
cmake_minimum_required(VERSION 3.28)
project(screensaver_p1 C)

set(CMAKE_C_STANDARD 17)
set(CMAKE_MODULE_PATH ${CMAKE_SOURCE_DIR}/cmake_modules)

set(SDL2_PATH "Sustituye tu ruta aquí")

find_package(SDL2 REQUIRED)
include_directories(${SDL2_INCLUDE_DIR})

add_executable(screensaver_p1 src/main.c
        src/config.h
        src/sweeping_line.h
        src/sweeping_line.c
        src/circle.h
        src/circle.c
        src/mystify_line.h
        src/mystify_line.c)


target_link_libraries(screensaver_p1 ${SDL2_LIBRARY})
```

> **INFO**
>
> Recuerda que luego de hacer el cambio en dicho archivo debes ejecutar el mismo, esto lo puedes hacer desde CLion



Parámetros configurables

- Número de líneas iniciales: es la cantidad inicial de entidades de tipo cuadrado/rectángulo
- Número de círculos iniciales: es la cantidad inicial de entidades de tipo círculo

Estos parámetros son los solicitados al usuario al ejecutar la aplicación, en caso de que no se proporcionen se usarán parámetros por defecto.

Para poder proporcionar estos valores de debe compilar el programa, siempre con CLion y luego desde la terminal usar el siguiente comando.

Ejecución del programa con terminal

```
./screensaver_p1.exe numLines numCircles
```

Comportamiento del Screensaver

El screensaver contiene 3 componentes principales los cuales son:

Mystify Lines

Las cuales son líneas que se mueven suavemente por la pantalla formando patrones geométricos y curvas elegantes. Estas se mueven en el fondo detrás de los demás componentes y colisionan con los bordes de la pantalla.

Círculos

Son círculos con tamaños y colores pseudoaleatorios, que se mueven con una velocidad pseudoaleatoria e interactuan con tanto con otros círculos como cuadrados y rectángulos, al colisionar estos tienen una probabilidad de generar otra instancia de la misma, la cuál se genera en una posición pseudoaleatoria, esto hasta que llega un valor máximo predeterminado para no afectar el rendimiento del programa.

Cuadrados y rectángulos

Son entidades con tamaños y colores pseudoaleatorios, que se mueven con una velocidad pseudoaleatoria e interactuan con tanto con otros círculos como cuadrados y rectángulos, al colisionar estos tienen una probabilidad de generar otra instancia de la misma, la cuál se genera en una posición pseudoaleatoria, esto hasta que llega un valor máximo predeterminado para no afectar el rendimiento del programa.

> **INFO**
>
> En la esquina superior izqierda se puede observar un contador con las entidades y la cantidad de FPS con las que se muestra la aplicación

Conclusiones

La Versión secuencial del screen saver muestra una mayor estabilidad y predictibilidad bajo condiciones de alta carga, con una disminución gradual en los FPS a medida que se incrementa el número de elementos. Tanto las líneas como los círculos tienden a alcanzar el límite máximo de 200 elementos después de 30 segundos, lo que indica una saturación predecible y consistente. Esta versión es más adecuada para situaciones en las que se espera una carga constante y alta, manteniendo un rendimiento más estable.

Por otro lado, la Versión paralela es más eficiente en condiciones de baja carga, manteniendo altos FPS cuando hay pocos elementos iniciales. Sin embargo, sufre caídas más abruptas en los FPS cuando aumenta el número de elementos, y su comportamiento es menos predecible, especialmente en lo que respecta a la cantidad de círculos dibujados. Esta versión podría ser más adecuada para escenarios donde se prioriza el rendimiento con baja carga, pero es menos estable bajo alta carga en comparación con la versión secuencial.

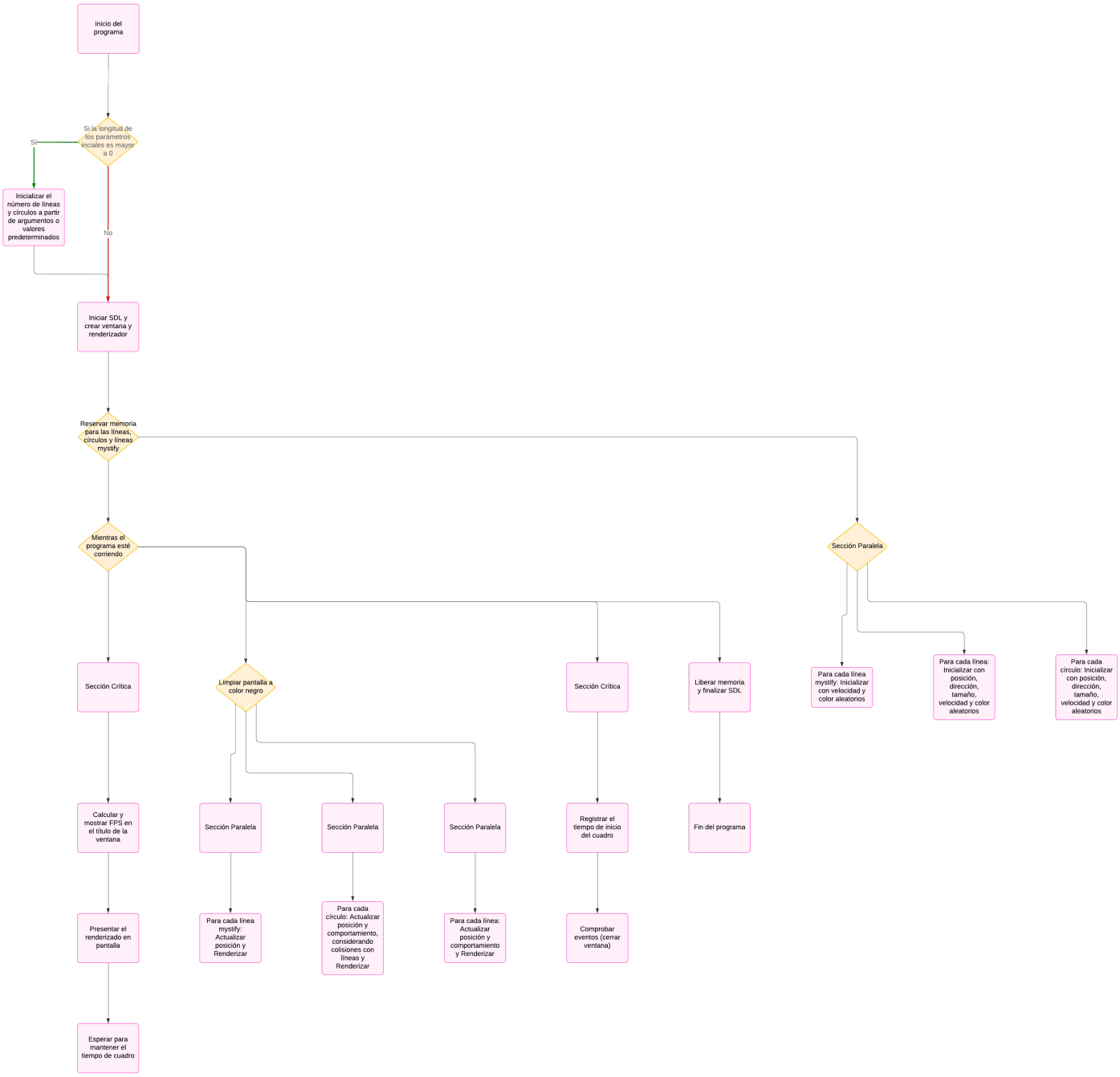
Recomendaciones

- Se recomienda el uso del IDE CLION de JetBrains debido a que el mismo realiza las configuraciones necesarias para la ejecución de programas en C.
- Se recomienda el uso de la paralelización con OpenMP para un mejor rendimiento.

Referencias

- Ritchie, D. M. (2003). The development of the C language. Bell Labs. Recuperado 29 de agosto de 2024, de https://www.bell-labs.com/usr/dmr/www/chist.html
- University of Notre Dame. (s. f.). Lecture 12: Introduction to OpenMP (Part 1). Recuperado 29 de agosto de 2024, de https://www3.nd.edu/~zxu2/acms60212-40212/Lec-12-OpenMP.pdf
- SDL. (2023). Simple DirectMedia Layer - Homepage. Recuperado 29 de agosto de 2024, de https://www.libsdl.org/

Anexo 1 - Diagrama de flujo del programa



Anexo 2 - Catálogo de funciones

init_circle

- Entrada
  - circle (Circle*): Puntero a la entidad de círculo (`Circle`) que se va a inicializar.
  - x (int): Coordenada en x de la posición inicial del círculo.
  - y (int): Coordenada en y de la posición inicial del círculo.
  - dx (int): Dirección en x del círculo.
  - dy (int): Dirección en y del círculo.
  - radius (int): Radio del círculo.
  - speed (int): Velocidad de movimiento del círculo.
  - color (SDL_Color): Color del círculo.
- Salida
  - Esta función no retorna valores.
- Descripción
  - Constructor que inicializa un círculo (`Circle`) con las propiedades especificadas como posición, dirección, radio, velocidad y color.

update_circle

- Entrada
  - circle (Circle): Entidad de círculo.
  - lines (SweepingLine): Lista de líneas con las que puede colisionar el círculo.
  - num_lines (int): Número de líneas en la lista.
  - circles (Circle): Lista de otros círculos para posibles interacciones.
  - num_circles (int): Número de círculos en la lista.
- Salida
  - (bool): Retorna `true` si el círculo debe ser eliminado, `false` de lo contrario.
- Descripción
  - Revisa el estado del círculo y actualiza su posición. Si colisiona con algo, se invierte su dirección, se crea un nuevo círculo.

render_circle

- Entrada
  - circle (Circle*): Puntero a la entidad de círculo que se va a renderizar.
  - renderer (SDL_Renderer*): Puntero al renderer donde se va a dibujar el círculo.
- Salida
  - Esta función no retorna valores.
- Descripción
  - Renderiza el círculo en el renderer especificado, utilizando las propiedades actuales del círculo como posición, tamaño y color.

init_sweeping_line

- Entrada
  - line (SweepingLine*): Puntero a la entidad de línea móvil (`SweepingLine`) que se va a inicializar.
  - x (int): Coordenada en x de la posición inicial de la línea.
  - y (int): Coordenada en y de la posición inicial de la línea.
  - dx (int): Dirección en x de la línea.
  - dy (int): Dirección en y de la línea.
  - length (int): Longitud de la línea.
  - width (int): Ancho de la línea.
  - speed (int): Velocidad de movimiento de la línea.
  - color (SDL_Color): Color de la línea.
- Salida
  - Esta función no retorna valores.
- Descripción
  - Constructor que inicializa una línea móvil (`SweepingLine`) con las propiedades especificadas como posición, dirección, longitud, ancho, velocidad y color.

update_sweeping_line

- Entrada
  - line (SweepingLine*): Puntero a la entidad de línea móvil (`SweepingLine`) que se va a actualizar.

    - lines (SweepingLine**): Doble puntero a una lista de líneas móviles con las que puede colisionar.
    - num_lines (int*): Puntero al número actual de líneas en la lista.
- Salida
    - (bool): Retorna `true` si la línea debe ser eliminada, `false` de lo contrario.
- Descripción
    - Revisa el estado de la línea y actualiza su posición. Si colisiona con algo, invierte su dirección y puede crear una nueva línea.

render_sweeping_line

- Entrada
    - line (SweepingLine*): Puntero a la entidad de línea móvil (`SweepingLine`) que se va a dibujar.
    - renderer (SDL_Renderer*): Puntero al renderer donde se va a renderizar la línea.
- Salida
    - Esta función no retorna valores.
- Descripción
    - Dibuja la línea móvil (`SweepingLine`) en la ventana utilizando el renderer especificado, basándose en las propiedades actuales de la línea como posición, longitud, ancho y color.

main

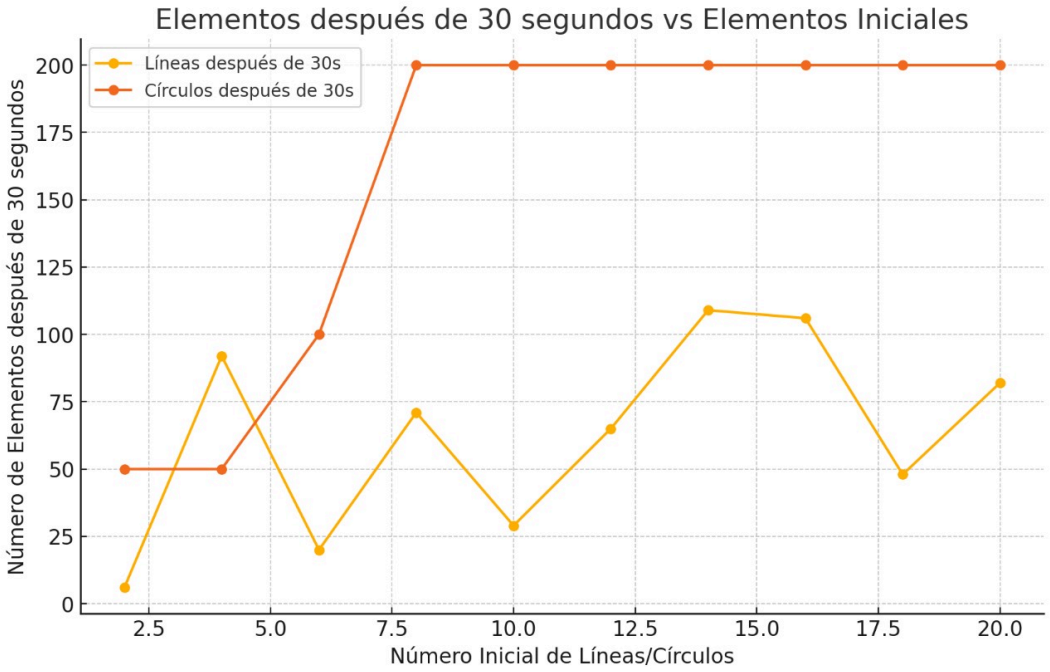
- Entrada
    - argc (int): Número de argumentos pasados por línea de comandos.
    - argv (char*[]): Array de cadenas de caracteres que contiene los argumentos pasados por línea de comandos.
- Salida
    - (int): Retorna un valor entero que indica el estado de finalización del programa (`0` para éxito).
- Descripción
    - Punto de entrada principal del programa. Inicializa los recursos necesarios, gestiona el bucle principal de la aplicación, y maneja la ejecución y finalización del programa basándose en los argumentos de entrada.

Anexo 3 - Bitácora de Pruebas

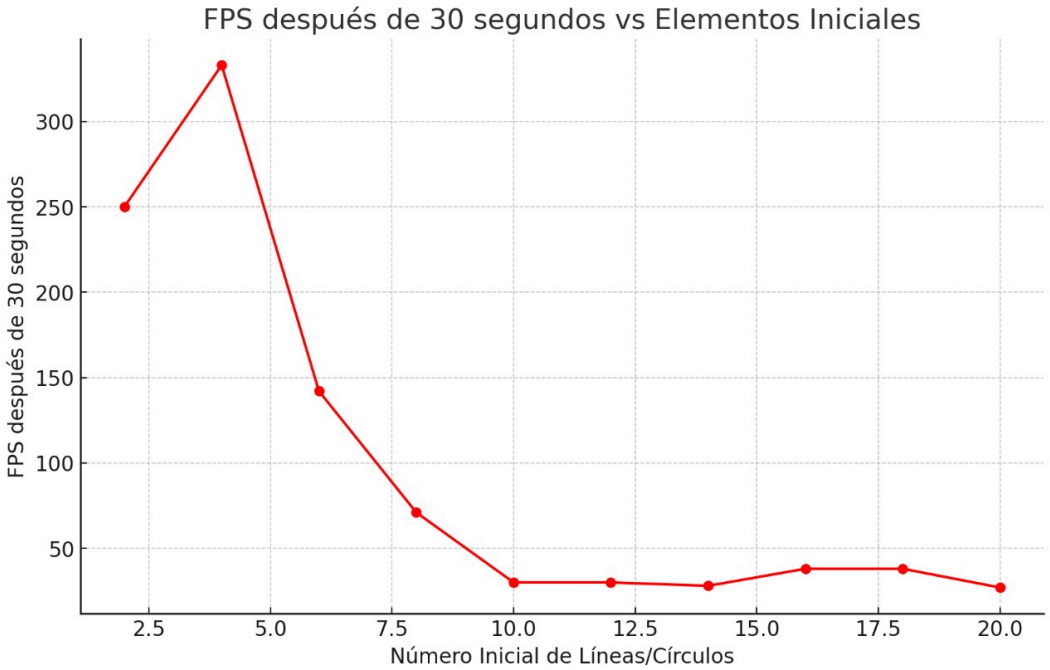
A continuación se muestran los resultados de las pruebas que se realizaron, tanto con la versión secuencial como con la paralela.

Secuencial

Líneas Iniciales	Círculos Iniciales	Líneas después de 30 segundos	Círculos después de 30 segundos	FPS después de 30 segundos
2	2	6	50	250
4	4	92	50	333
6	6	20	100	142
8	8	71	200	71
10	10	29	200	30
12	12	65	200	30
14	14	109	200	28
16	16	106	200	38
18	18	48	200	38
20	20	82	200	27



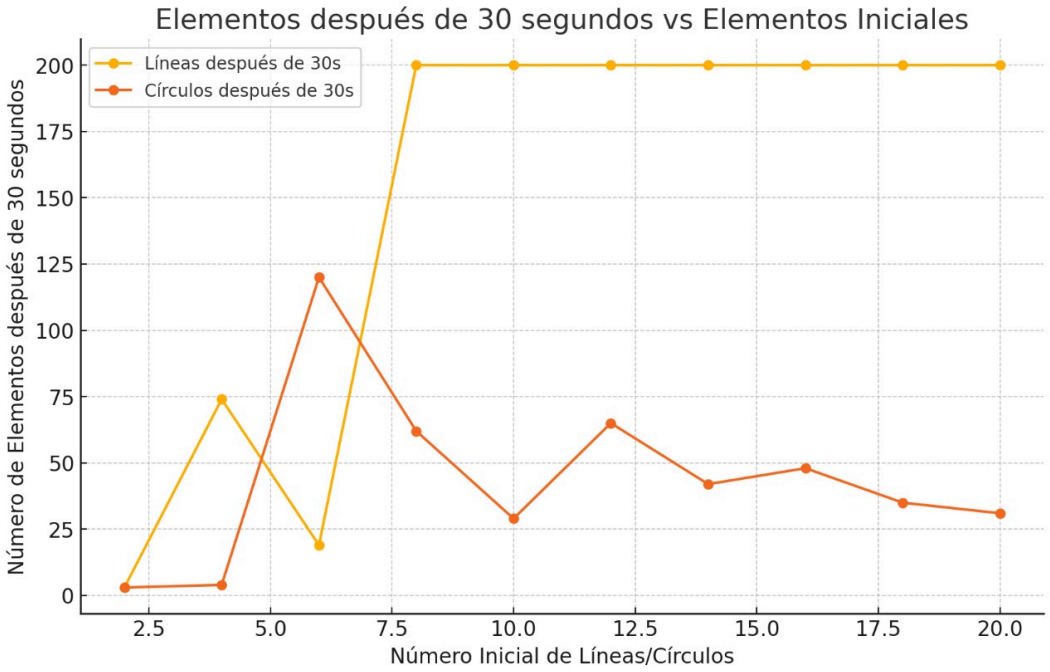
Muestra el número de líneas y círculos después de 30 segundos en función del número inicial de líneas y círculos. Podemos observar que el número de círculos rápidamente alcanza el límite de 200, independientemente del número inicial, mientras que las líneas muestran un comportamiento más variable.



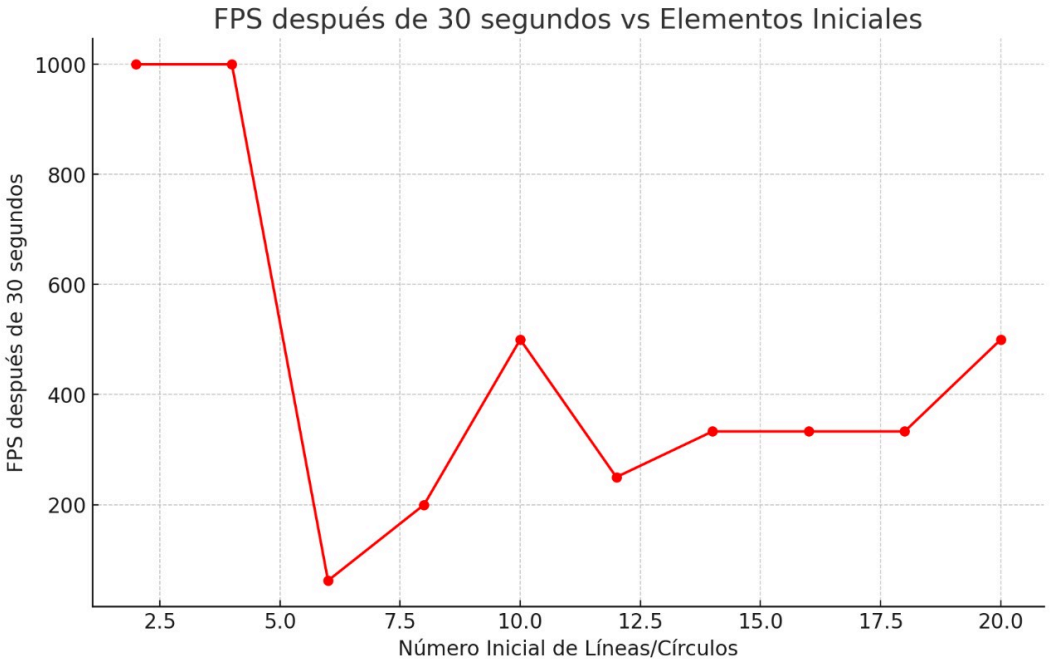
Representa los FPS después de 30 segundos en función del número inicial de líneas y círculos. A medida que aumentan los elementos iniciales, los FPS disminuyen significativamente, lo que indica una sobrecarga en el sistema, especialmente cuando se aproxima al máximo permitido de elementos.

Paralela

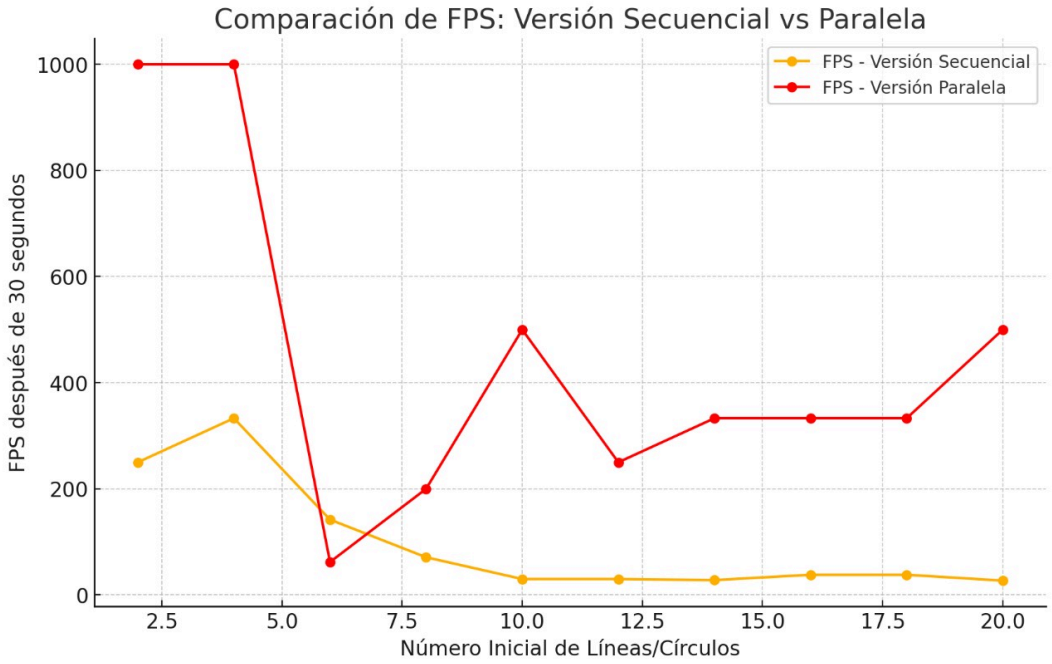
Líneas Iniciales	Círculos Iniciales	Líneas después de 30 segundos	Círculos después de 30 segundos	FPS después de 30 segundos
2	2	3	3	1000
4	4	74	4	1000
6	6	19	120	62
8	8	200	62	200
10	10	200	29	500
12	12	200	65	250
14	14	200	42	333
16	16	200	48	333
18	18	200	35	333
20	20	200	31	500



Muestra el número de líneas y círculos después de 30 segundos en función del número inicial de líneas y círculos. Se observa que las líneas alcanzan rápidamente el límite máximo de 200, mientras que los círculos muestran más variabilidad y no siempre llegan al límite.



Representa los FPS después de 30 segundos en función del número inicial de líneas y círculos. Los FPS se mantienen altos (1000) con pocos elementos iniciales, pero disminuyen drásticamente a medida que se incrementa el número de elementos y se acerca al límite máximo permitido.



Representa la comparación entre el rendimiento en fps de la versión secuencial y paralela del proyecto. Se observa una clara tendencia en favor de la versión paralela, teniendo signficativamente más fps en todas las pruebas.

Edit this page