Universidad del Valle de Guatemala

Computación Paralela y Distribuída

Catedrático: Sebastián Galindo

Sección 10

Proyecto #1

Brandon Ronaldo Sicay Cumes 21757

Daniel Esteban Morales Urizar 21785

Guatemala, 18 de agosto 2024

Descripción del proyecto

Este proyecto simula un screensaver en el que varios sprites (pokemones) y una pokebola se mueven de manera continua por la pantalla, rebotando entre los bordes y entre ellos mismos. La pokebola tiene la capacidad de "capturar" pokémons cuando colisionan con ellos, haciendo que desaparezcan de la pantalla. La simulación está diseñada para ejecutarse tanto en versión serial como paralela, con la finalidad de comparar el rendimiento y calcular el speedup.

Cálculos realizados

El movimiento de los pokemones y de la pokebola está basado en cálculos trigonométricos sencillos. Cada sprite tiene una velocidad y un ángulo de movimiento, que se calculan al inicio de la simulación. A lo largo del tiempo, los pokemones y la pokebola se mueven según las fórmulas de desplazamiento

$$pokemon -> position.x \ += \ pokemon -> speed \ * \ cos(pokemon -> angle)$$
$$pokemon -> position.y \ += \ pokemon -> speed \ * \ cos(pokemon -> angle)$$

Además, se implementaron rebotes en los bordes de la pantalla y colisiones entre los pokemones. Cuando un pokémon colisiona con otro, se intercambian sus ángulos de movimiento, simulando un rebote.

Paralelización

La paralelización del proyecto se centró en las partes más costosas computacionalmente, específicamente en el cálculo del movimiento y las colisiones entre los pokemones. Las funciones que mueven a cada pokemon y las que gestionan sus colisiones fueron paralelizadas usando OpenMP. Estas se paralelizaron debido a la naturaleza independiente de las operaciones: cada pokémon puede moverse de manera independiente, y las colisiones entre pokémons también se pueden manejar de forma eficiente dentro de un bucle paralelo.

La finalidad de la paralelización fue acelerar el cálculo del movimiento de los pokemones, reducir el tiempo total de procesamiento en cada frame, aprovechar los múltiples núcleos de CPU para realizar las operaciones más intensivas de manera simultánea, mejorando así el rendimiento global de la simulación.

Cálculos de tiempo y eficiencia

Para medir el tiempo de ejecución del programa y posteriormente calcular el speedup, se utilizó una metodología basada en la medición del tiempo por cada frame de ejecución. El tiempo se captura antes y después del procesamiento de cada frame utilizando una función que permite medir con precisión el tiempo en ciclos de reloj del sistema. Se acumula el tiempo total que toma ejecutar el procesamiento de todos los elementos en cada frame, y tras 100 iteraciones, se calcula el tiempo promedio por iteración dividiendo el tiempo total acumulado entre el número de iteraciones.

Este tiempo promedio por iteración proporciona una medición estable del tiempo que tarda el programa en procesar un frame completo de la simulación, lo que incluye el movimiento de los pokemones, la detección de colisiones y otros cálculos relacionados. A partir de este tiempo promedio, se puede comparar la versión serial con la versión paralela del programa para calcular el speedup.

Ejecución con 250 sprites

```
danielmo@danielmo-VirtualBox:/media/sf_labs/proyeto1PP$ ./screensaver 250
Tiempo promedio de ejecución para 100 iteraciones: 41.26 ms
Tiempo promedio de ejecución para 100 iteraciones: 40.77 ms
Tiempo promedio de ejecución para 100 iteraciones: 39.22 ms
Tiempo promedio de ejecución para 100 iteraciones: 36.23 ms
Tiempo promedio de ejecución para 100 iteraciones: 37.47 ms
```

```
danielmo@danielmo-VirtualBox:/media/sf_labs/proyeto1PP$ ./screensaverOM 250
Tiempo promedio de ejecución para 100 iteraciones: 39.05 ms
Tiempo promedio de ejecución para 100 iteraciones: 37.86 ms
Tiempo promedio de ejecución para 100 iteraciones: 40.17 ms
Tiempo promedio de ejecución para 100 iteraciones: 35.84 ms
Tiempo promedio de ejecución para 100 iteraciones: 36.02 ms
```

Tiempo promedio versión serial = 38.99 ms
Tiempo promedio versión paralela = 37.78 ms
SpeedUp = 1.03

Ejecución con 50 sprites

```
danielmo@danielmo-VirtualBox:/media/sf_labs/proyeto1PP$ ./screensaver 50
Tiempo promedio de ejecución para 100 iteraciones: 28.80 ms
Tiempo promedio de ejecución para 100 iteraciones: 29.75 ms
Tiempo promedio de ejecución para 100 iteraciones: 28.28 ms
Tiempo promedio de ejecución para 100 iteraciones: 29.08 ms
Tiempo promedio de ejecución para 100 iteraciones: 29.96 ms
```

```
danielmo@danielmo-VirtualBox:/media/sf_labs/proyeto1PP$ ./screensaverOM 50
Tiempo promedio de ejecución para 100 iteraciones: 30.69 ms
Tiempo promedio de ejecución para 100 iteraciones: 29.59 ms
Tiempo promedio de ejecución para 100 iteraciones: 29.69 ms
Tiempo promedio de ejecución para 100 iteraciones: 26.89 ms
Tiempo promedio de ejecución para 100 iteraciones: 27.06 ms
```

Tiempo promedio versión serial = 29.17 ms
Tiempo promedio versión paralela = 28.78 ms
SpeedUp = 1.01

Ejecución con 500 sprites

```
danielmo@danielmo-VirtualBox:/media/sf_labs/proyeto1PP$ ./screensaver 500
Tiempo promedio de ejecución para 100 iteraciones: 52.65 ms
Tiempo promedio de ejecución para 100 iteraciones: 50.54 ms
Tiempo promedio de ejecución para 100 iteraciones: 52.01 ms
Tiempo promedio de ejecución para 100 iteraciones: 48.27 ms
```

```
danielmo@danielmo-VirtualBox:/media/sf_labs/proyeto1PP$ ./screensaverOM 500
Tiempo promedio de ejecución para 100 iteraciones: 51.72 ms
Tiempo promedio de ejecución para 100 iteraciones: 51.08 ms
Tiempo promedio de ejecución para 100 iteraciones: 49.11 ms
Tiempo promedio de ejecución para 100 iteraciones: 46.73 ms
```

Tiempo promedio versión serial = 50.86 ms
Tiempo promedio versión paralela = 49.66 ms
SepeedUp = 1.02

Ejecución con 250 sprites

```
danielmo@danielmo-VirtualBox:/media/sf_labs/proyeto1PP$ ./screensaver 250
Tiempo promedio de ejecución para 100 iteraciones: 42.93 ms
Tiempo promedio de ejecución para 100 iteraciones: 41.55 ms
Tiempo promedio de ejecución para 100 iteraciones: 42.31 ms
Tiempo promedio de ejecución para 100 iteraciones: 42.82 ms
Tiempo promedio de ejecución para 100 iteraciones: 42.29 ms
Tiempo promedio de ejecución para 100 iteraciones: 40.75 ms
Tiempo promedio de ejecución para 100 iteraciones: 39.80 ms
Tiempo promedio de ejecución para 100 iteraciones: 48.67 ms
Tiempo promedio de ejecución para 100 iteraciones: 56.60 ms
Tiempo promedio de ejecución para 100 iteraciones: 48.58 ms
Tiempo promedio de ejecución para 100 iteraciones: 46.87 ms
Tiempo promedio de ejecución para 100 iteraciones: 49.51 ms
Tiempo promedio de ejecución para 100 iteraciones: 54.05 ms
Tiempo promedio de ejecución para 100 iteraciones: 50.82 ms
Tiempo promedio de ejecución para 100 iteraciones: 43.13 ms
Tiempo promedio de ejecución para 100 iteraciones: 41.55 ms
Tiempo promedio de ejecución para 100 iteraciones: 40.46 ms
```

```
danielmo@danielmo-VirtualBox:/media/sf_labs/proyeto1PP$ ./screensaverOM 250
Tiempo promedio de ejecución para 100 iteraciones: 47.28 ms
Tiempo promedio de ejecución para 100 iteraciones: 43.56 ms
Tiempo promedio de ejecución para 100 iteraciones: 42.83 ms
Tiempo promedio de ejecución para 100 iteraciones: 40.89 ms
Tiempo promedio de ejecución para 100 iteraciones: 41.57 ms
Tiempo promedio de ejecución para 100 iteraciones: 38.29 ms
Tiempo promedio de ejecución para 100 iteraciones: 36.39 ms
Tiempo promedio de ejecución para 100 iteraciones: 38.50 ms
Tiempo promedio de ejecución para 100 iteraciones: 34.99 ms
Tiempo promedio de ejecución para 100 iteraciones: 34.34 ms
Tiempo promedio de ejecución para 100 iteraciones: 35.36 ms
Tiempo promedio de ejecución para 100 iteraciones: 36.90 ms
Tiempo promedio de ejecución para 100 iteraciones: 35.82 ms
Tiempo promedio de ejecución para 100 iteraciones: 36.28 ms
Tiempo promedio de ejecución para 100 iteraciones: 35.49 ms
Tiempo promedio de ejecución para 100 iteraciones: 35.51 ms
Tiempo promedio de ejecución para 100 iteraciones: 37.38 ms
```

Tiempo promedio versión serial = 42.46
Tiempo promedio versión paralela = 38.29
SepeedUp = 1.10

Conclusiones

La paralelización aplicada al cálculo del movimiento de los sprites (Pokémones y Pokeball) mostró una leve mejora en los tiempos de ejecución, aunque no resultó en un incremento sustancial en la eficiencia.

Dado que el movimiento de los sprites es una tarea con una carga computacional constante, este fue un buen candidato para la paralelización, permitiendo un balance de trabajo más efectivo entre los hilos.

Los SpeedUp obtenidos en todas las pruebas fueron cercanos a 1, con valores entre 1.01 y 1.10. Estos valores indican que la paralelización trajo mejoras, pero estas son modestas, lo que sugiere que el problema no es lo suficientemente intensivo en cálculos como para obtener grandes beneficios con la paralelización a este nivel.

Repositorio

https://github.com/bsicay/proyeto1PP