Ant Simulator

Diagnóstico de optimización

Catania, Gian Franco

Nos encontramos con un proyecto ineficiente, encontrándonos con un mundo de hormigas que buscan comida, dejan feromonas y vuelven a su base a dejar las provisiones para luego seguir acumulando.

Se comenzó observando el juego en funcionamiento antes de ver el código fuente para verlo paso a paso. Al ejecutarse se observa que las hormigas consumen el alimento y vuelven a su base dejando feromonas. Los alimentos no se van destruyendo y se sigue generando más comida cada tiempo.

No se pudo lograr apreciar si las hormigas perdidas van directo a las feromonas por que se empieza a estar inestable para ver bien los procesos de lo que ocurre.

Se comenzó a leer el código fuente y dividiendo a cada objeto para encontrar paso a paso en los momentos que genera un mal rendimiento.

Se inició dejando solo a las hormigas deambulando con su nicho y se observa que no hay aumento del insecto, quedando siempre en una cantidad de 76 unidades. Se logra apreciar un rendimiento fenomenal.

El siguiente paso fue verificar el caso de la comida. Se comprueba que las hormigas no están consumiendo y se sigue generando de más.

Por último, las feromonas no se encontró un problema de funcionamiento, solo se comprueba que las hormigas no las detectan por lo que se sigue acumulando, generando una inestabilidad. Esto da a entender que estas consumen recursos de rendimiento.

Entre la lectura, pruebas y sin requerir de una herramienta, se lanza una hipótesis de que el problema central son los métodos que se encuentran en el chequeo de las comidas y feromonas que lo encontramos en la clase **Ant**.

```
public override void UpdateOn(World world)
{
    if (hasFood)
    {
        Color = Color.Red;
        ReleasePheromone(world);
        MoveToNest(world);
    }
    else
    {
        Color = Color.Blue;
        Wander(world);
        if (!CheckFood(world))
        {
            CheckPheromone(world);
        }
    }
}
```

Se realiza la comprobación del rendimiento con la herramienta que ofrece Visual Studio y nos ofrece dos casos que están provocando una suma de recursos importante. Los dos se encuentran en la clase World.

En el primer caso se encarga de retornar el objeto con el valor que devuelve la función pura Dist. Si su distancia es menor a la distancia establecida, entonces se obtiene el GameObjects.

En el segundo caso se realiza un cálculo matemático que consiste en sacar la distancia entre dos puntos en las dos coordenadas. Potenciando a cada resultado a la 2, sumando luego los dos y finalizando con una fórmula que es la raíz cuadrada.