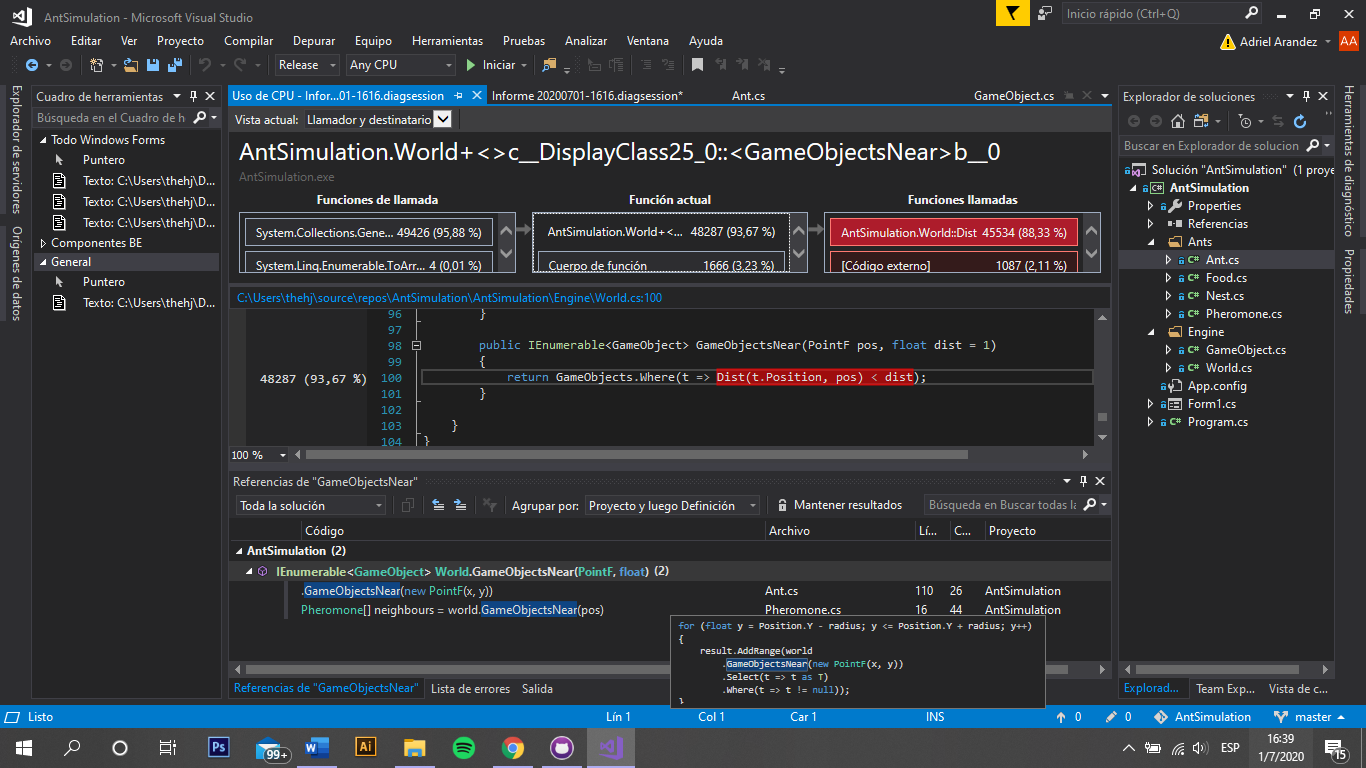
TP Parcial N°2

Técnicas de optimización

Arández Adriel

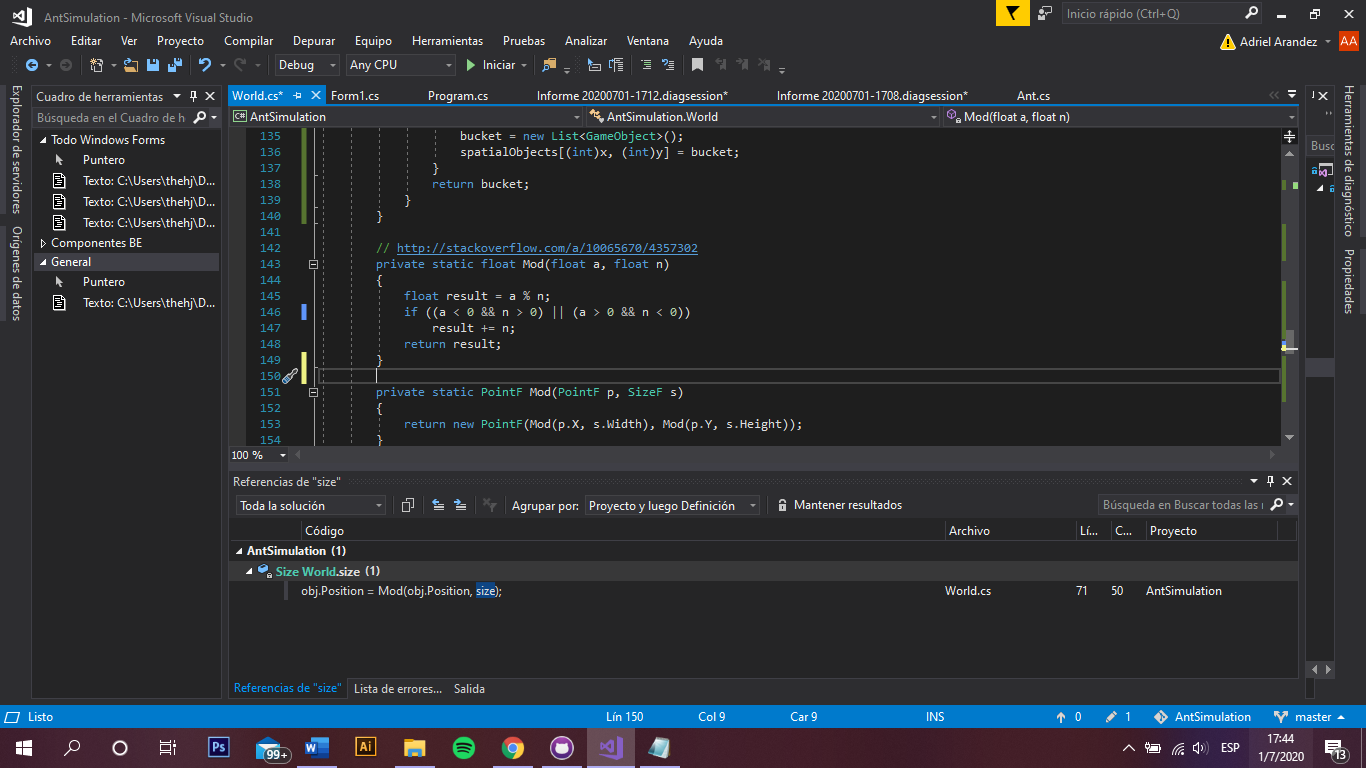
**Primera medición:**



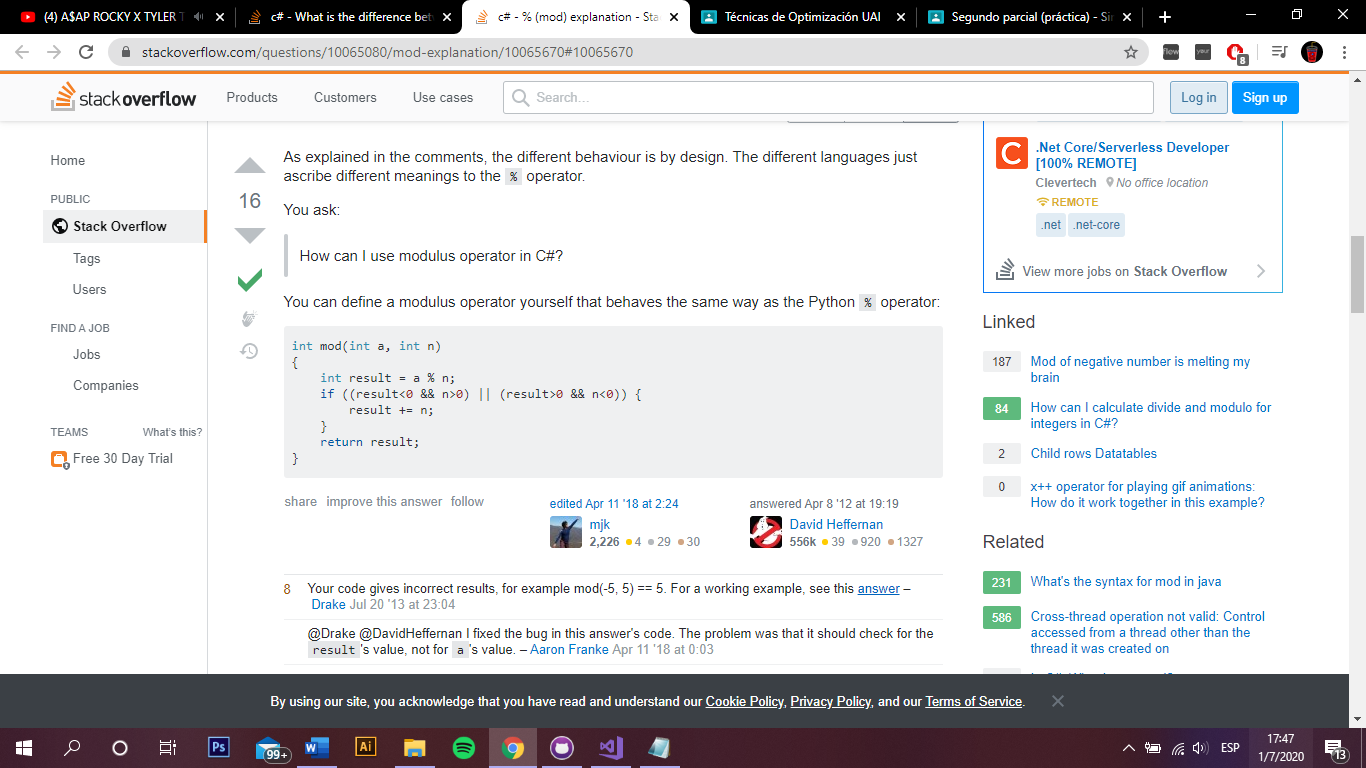
En la primera medición es visible como el sistema que se encarga de buscar por la comida de las hormigas y sobre todo el cálculo de distancia consume mucho procesamiento. Reemplazar el análisis total de los objetos por una spatial partition, analizando solo los cercanos debería funcionar mucho mejor.

**Aclaración:**

Durante la implementación de la spatial partition, tal y como se mencionó en la hipótesis, al ejecutar el programa y comprobar su comportamiento muchas veces fallaba ya que había objetos que se generaban o se desplazaban fuera de los límites de el “mundo” retornando *null* luego de analizar el código se encontró un error en la escritura de el método Mod:



El cual de acuerdo con la fuente de donde se extrajo debe escribirse de la siguiente manera:

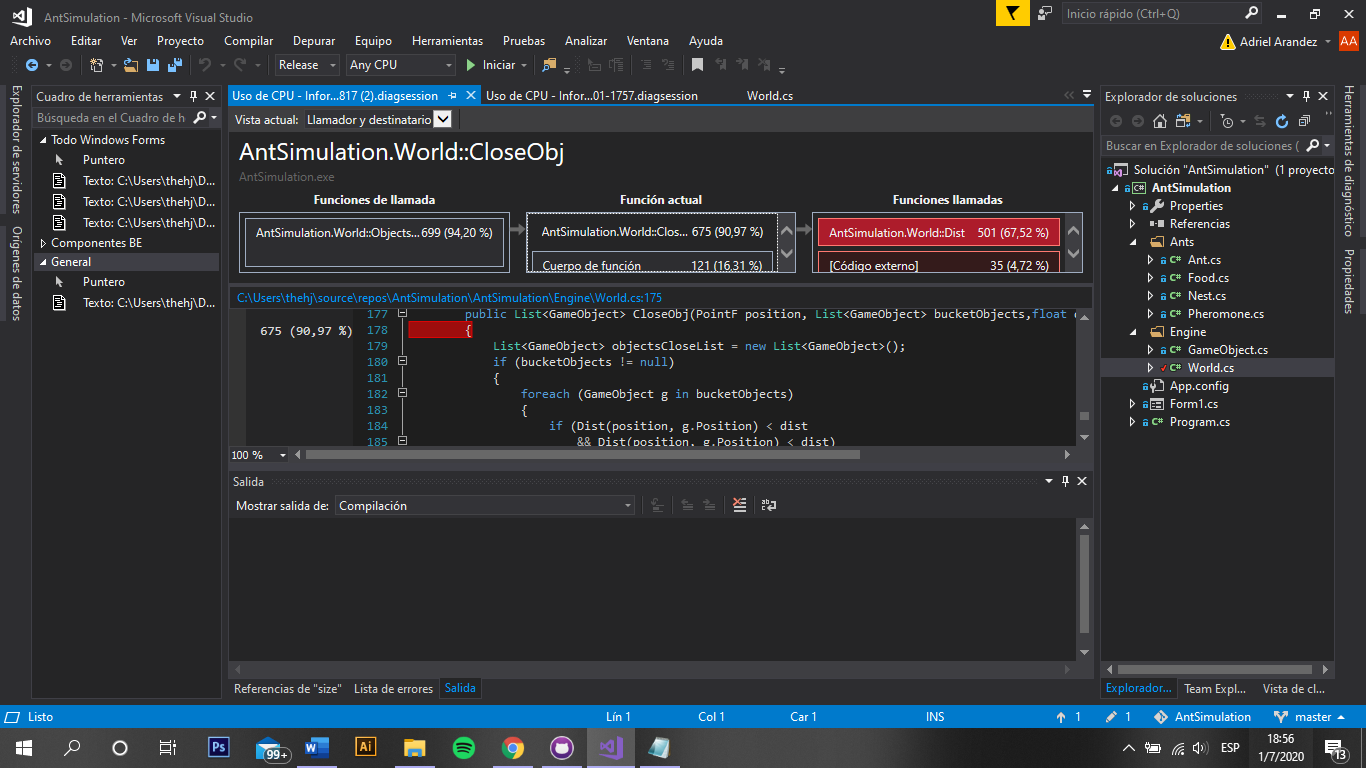


Esto en definitiva causaba que los objetos salieran de los límites y por eso retornaban null. Si bien se podría considerar un cambio de funcionamiento en el programa, en el video de muestra otorgado se ve claramente como una “comida de hormiga” excede su limite en la zona superior apareciendo la parte faltante en su correspondiente inferior: 

**Validación:**

Luego de aplicar la spatial partition el programa tuvo una mejor performance y aumentó visiblemente la cantidad de frames por segundo. La aplicación se traba cuando se generan muchas feromonas en pantalla.

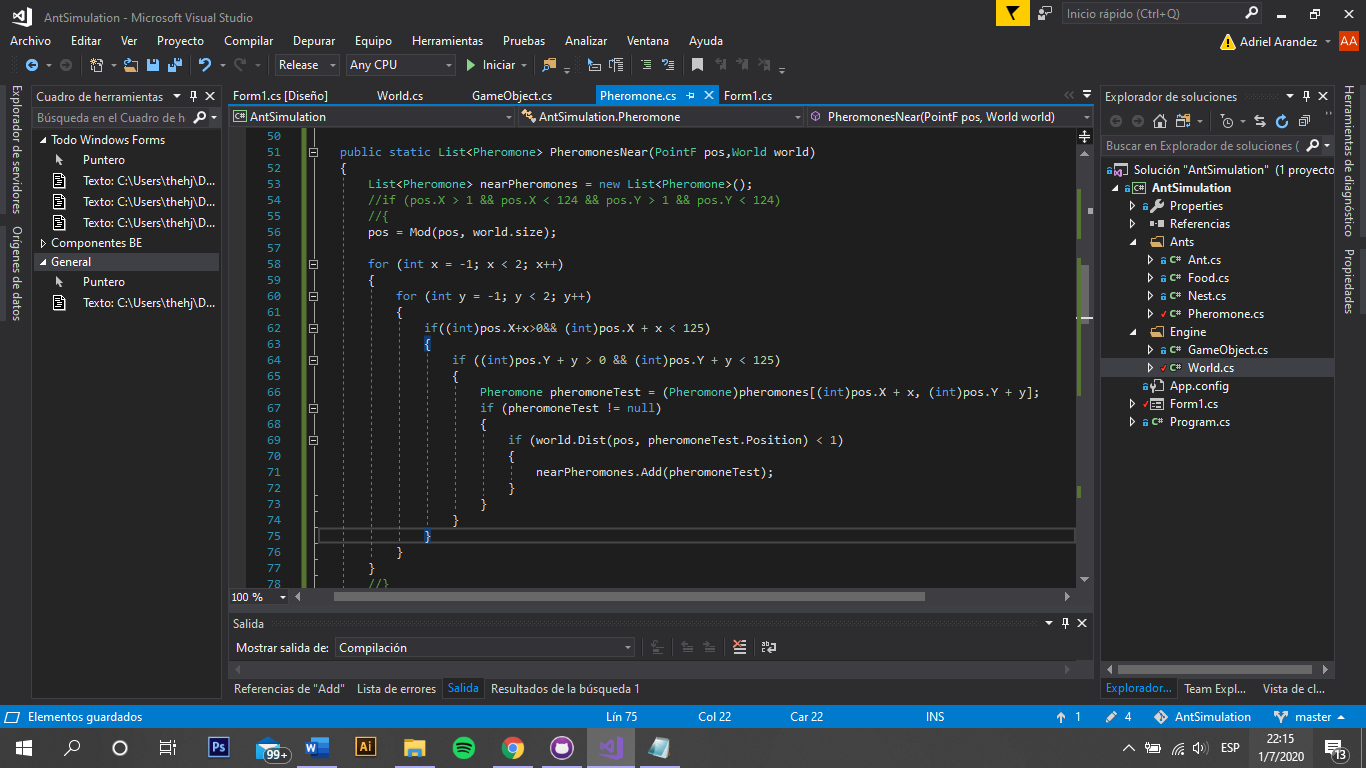
**Segunda medición:**

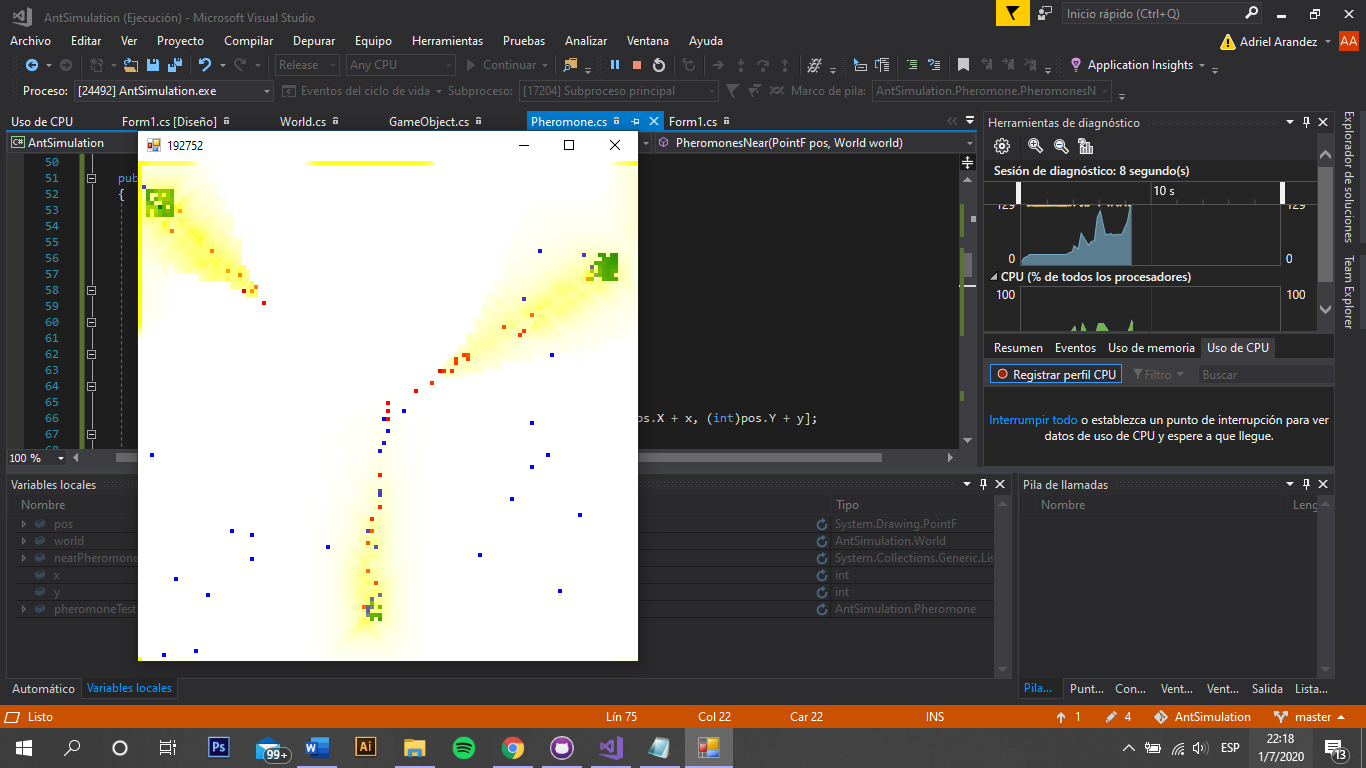


En la segunda medición realizada se pudo ver que el método que calculaba la distancia ocupaba muchos recursos, esto es causado por la clase de las feromonas, las cuales calculaban la distancia muchas veces. Para reducir esto lo lógico sería verificar el funcionamiento de las feromonas y ver cada cuanto y de que manera calculan esta distancia. En base a esto cambiar la distancia de la spatial partition podría ser una solución.

**Resolución:**

La primera solución encarada para solucionar el problema fue cambiar el tamaño de la spatial partition, la hipótesis era que al hacerlo se reduciría la cantidad de cálculos de distancia, pero no fue así. Entonces, para optimizar estos cálculos se debía tener en cuenta únicamente a las feromonas y a su distancia, en base a esto al principio se realizó una lista estática de feromonas, la cual se recorrería para calcular la distancia entre ellas. Este método continuaba consumiendo más recursos de los necesarios. Lo que se decidió hacer fue convertir esta lista a una matriz de GameObjects en la cual se recorrerían las casillas adyacentes de cada feromona y así se calcularía la distancia entre ellas. Luego de solventar varios errores que colocaban feromonas fuera de la matriz el problema fue resuelto.



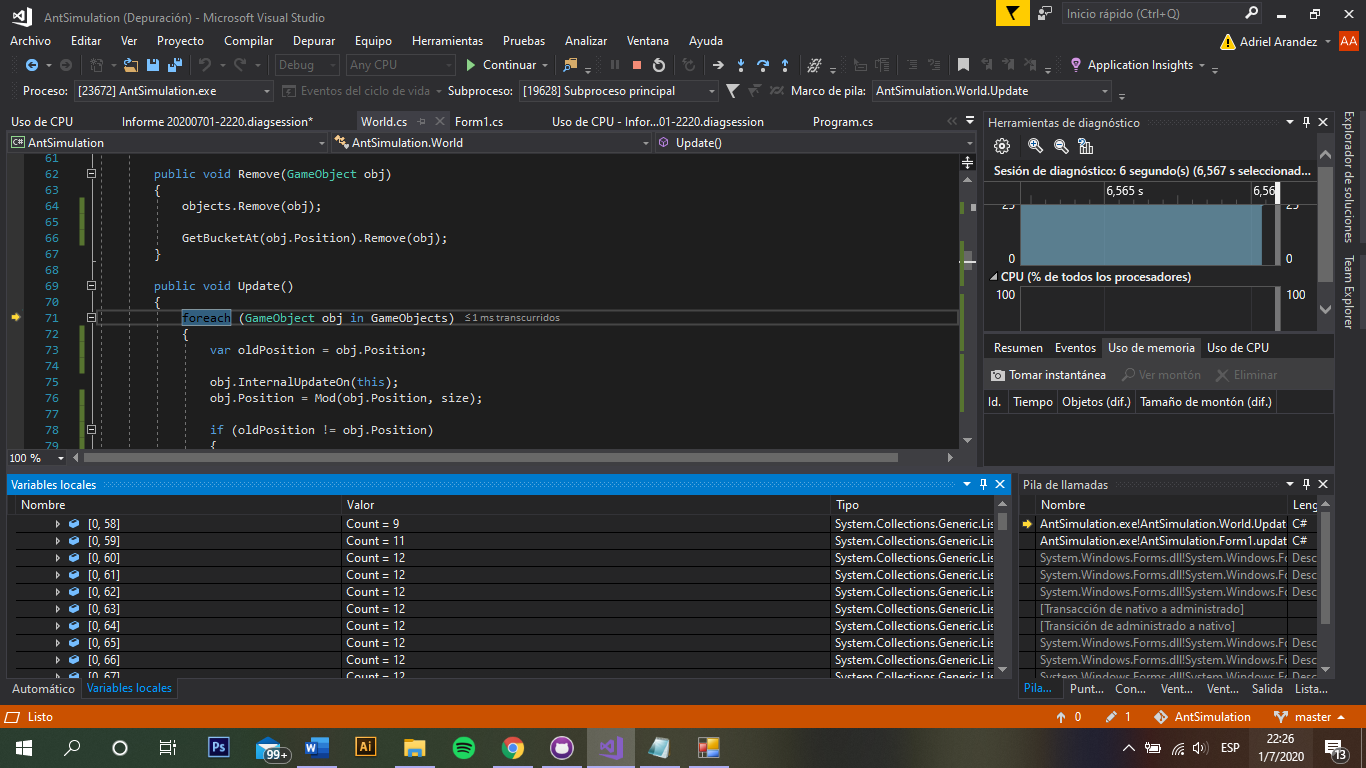
Ilustración -Método

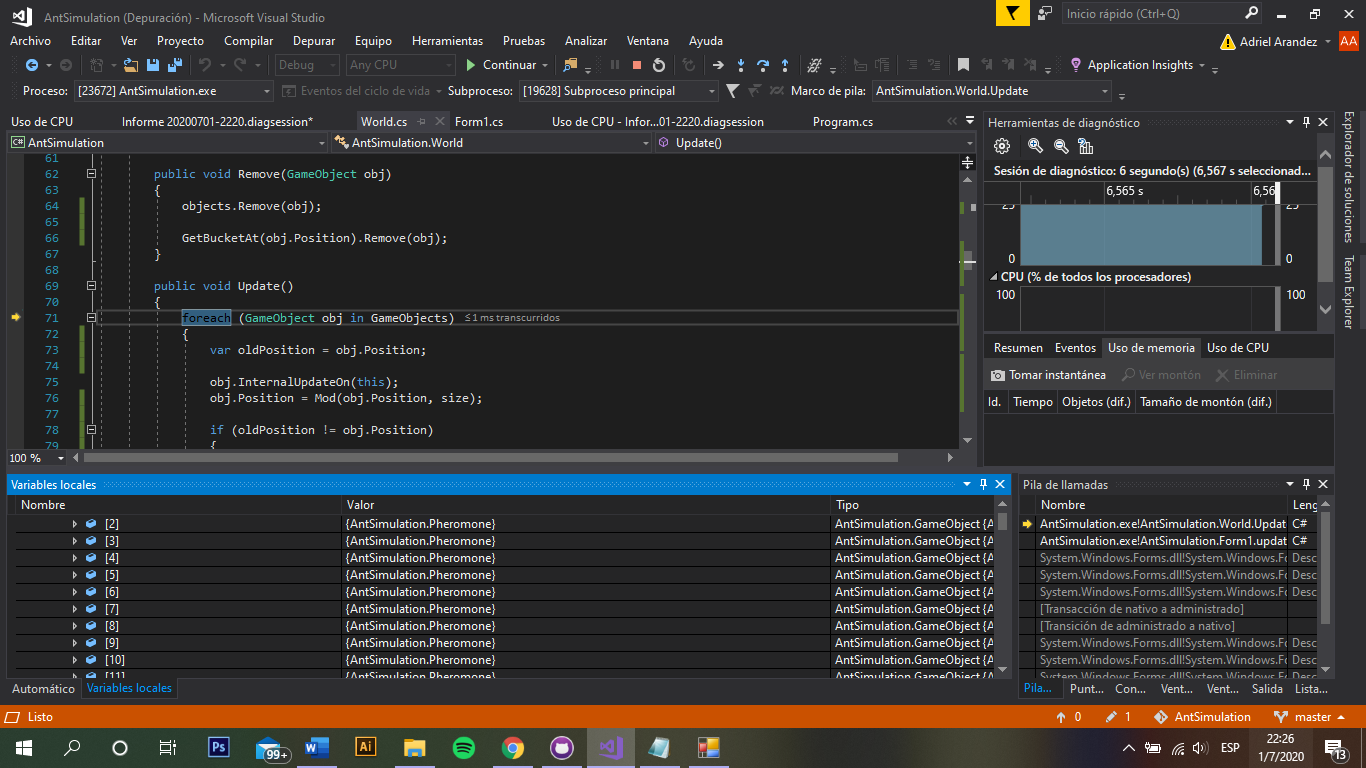
**Validación:**

Ahora la aplicación corre a mas cuadros por segundo, y si bien se sigue trabando, ahora lo hace cuando se llega a una mayor cantidad de elementos en pantalla.

**Tercera medición:**

En esta medición se verificó el motivo por el cual el programa se trababa, en las capturas se puede ver que existe una gran cantidad de instancias por lo que la generación y actualización de cada una de ellas puede ser el causante.

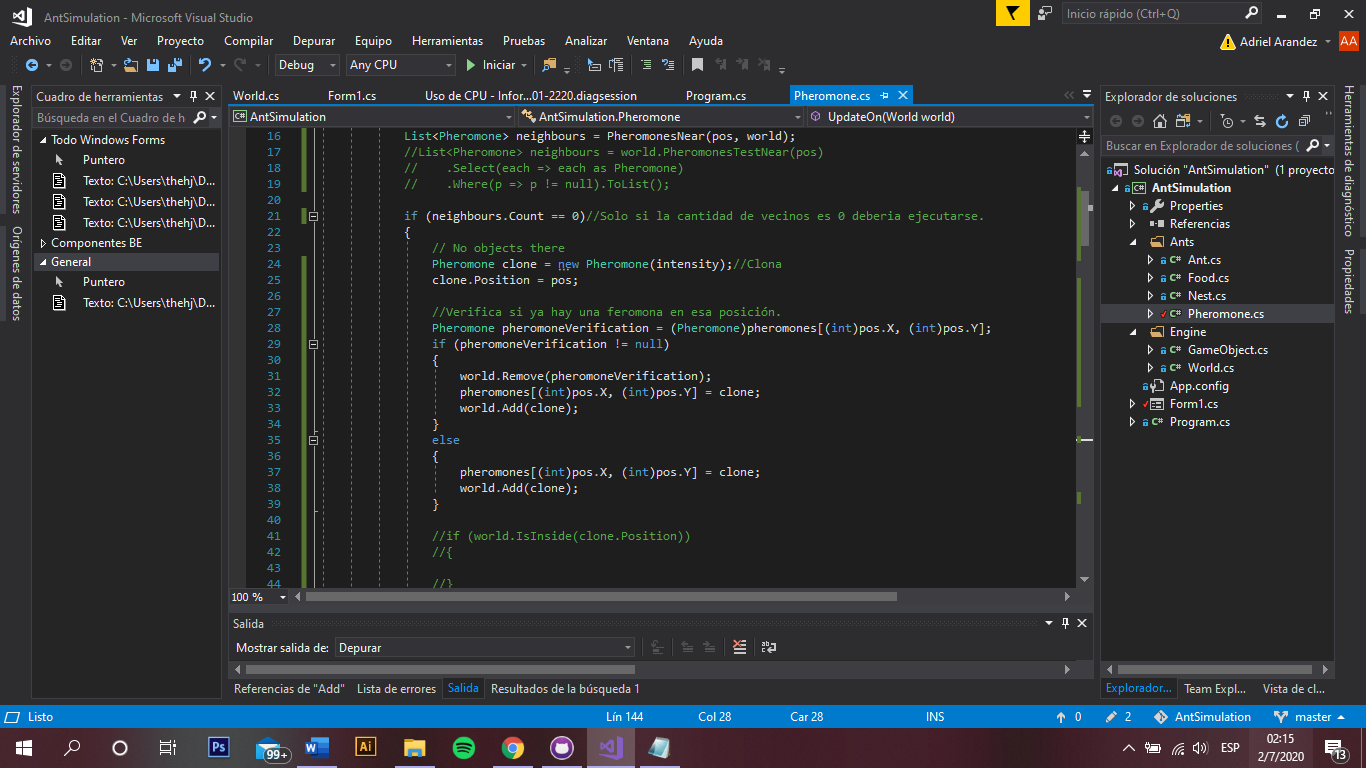




Además, en cada posición se acumulan más de una feromona. Limitar esto a una feromona por posición será probablemente la manera más optima de solucionarlo.

**Resolución:**

Tal y como se mencionó, limitar a una feromona por posición solucionó el problema.



**Validación:**

Ahora el programa se ejecuta manteniendo la cantidad de instancias y sin trabarse en ningún momento.

**Cuarta medición:**

En esta cuarta medición se ve que el método CheckFood ocupa gran parte del procesamiento, sin embargo el código funciona bien y esta suficientemente optimizado.

