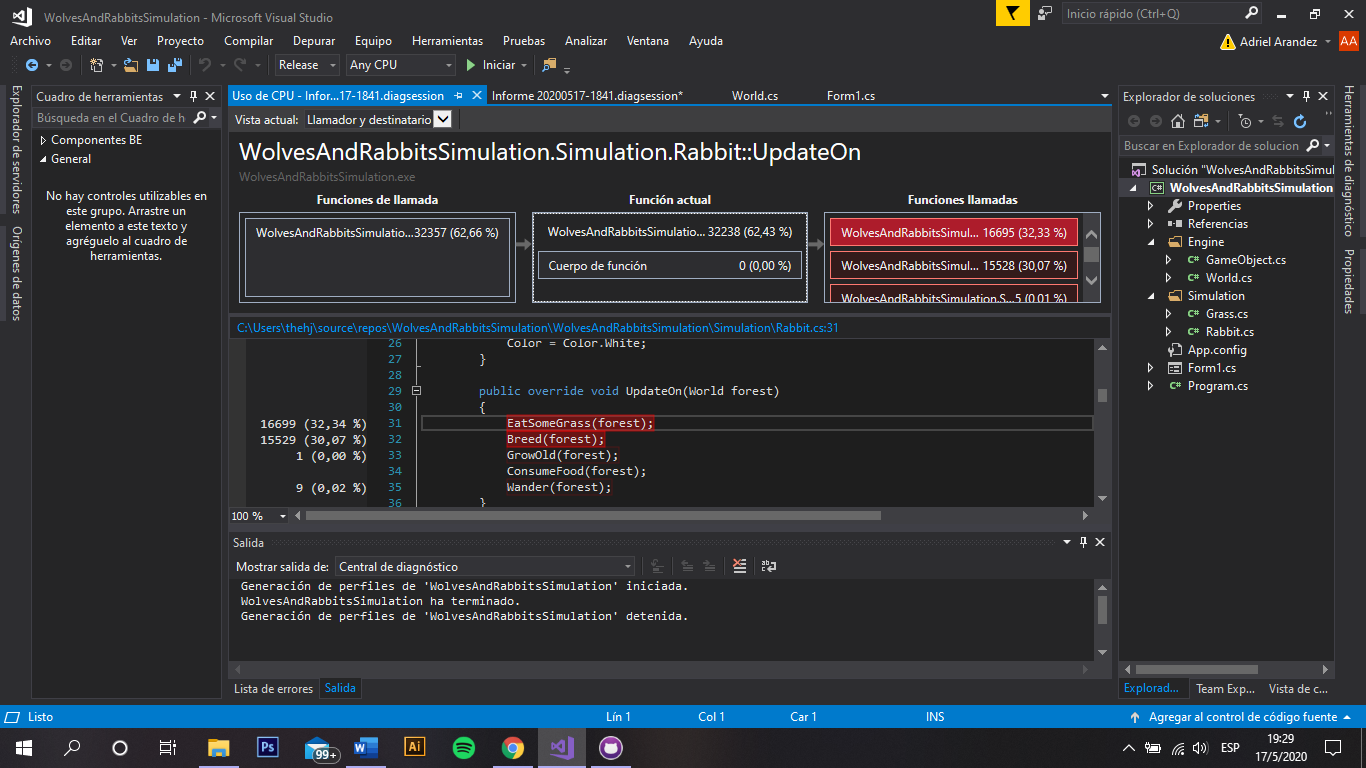
Primera medición:

La primera medición realizada muestra que el método EatSomeGrass y Breed acaparan gran parte del uso del CPU. En ambos casos se indica que el proceso de Linq utilizado está demorando más de lo esperado. Hipotéticamente si esto es modificado debería mejorar el funcionamiento.

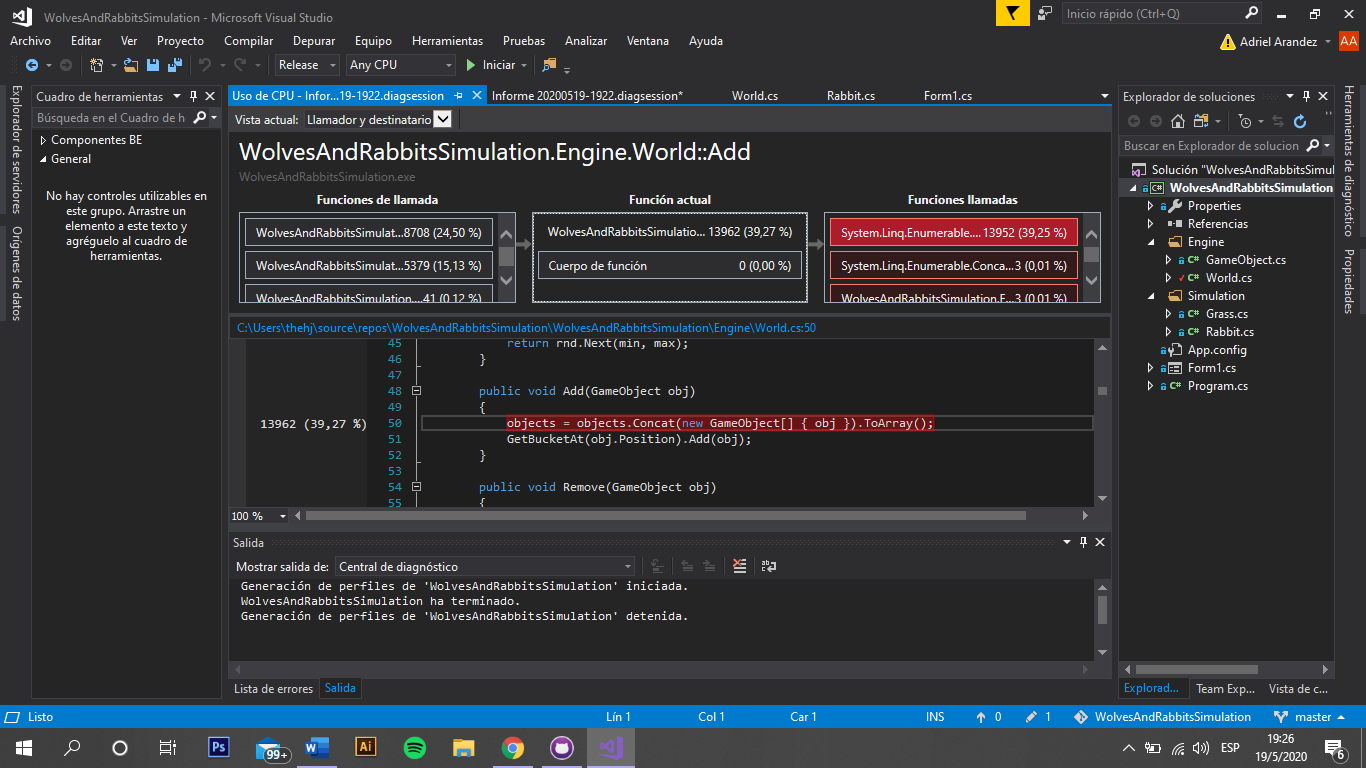


Resolución:

Para solucionar este problema fue aplicada la técnica de spatial partition utilizada más que nada para resolver el problema que generaba el anterior método ObjecsAt en el cual se sostenían los procedimientos de Linq que retrasaban el programa. Al principio sin tener en cuenta a los conejos y reproducción se creo una grilla basada en los espacios que ocupaba el pasto, cada bloque de pasto era una fracción de esta grilla. Funcionó perfectamente y el performance aumentó, aunque posteriormente se descubrió que esta nueva organización atentaba contra el método Breed que permitía que los conejos se reprodujeran. En base a esto se creó un método diferente al ObjecsAt que permitiera determinar que otros GameObjects se encuentran cercanos a una posición señalada, recorriendo la celda correspondiente de la grilla y las adyacentes.

Con esto realizado la performance aumentó y se mantuvo la funcionalidad.

Segunda medición:

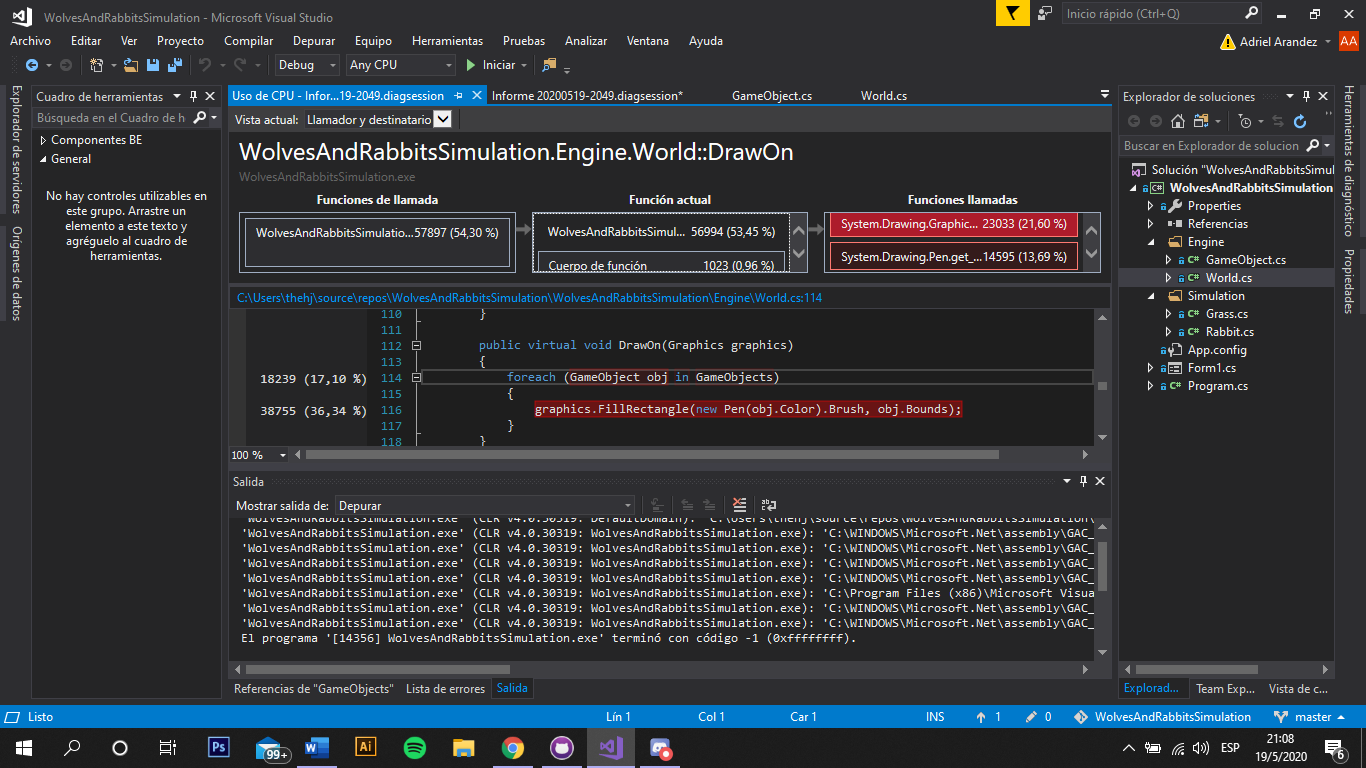


Al realizar una segunda medición el programa corría a 24 fps los primeros segundos, cuando los conejos comenzaron a reproducirse masivamente los frames disminuyeron, ubicándonos en el momento de ejecución se puede notar que el punto conflictivo esta ubicado al añadir los objetos al array *objects*, en esta línea de código podemos notar que no se añade simplemente el objeto sino que se crea uno nuevo el cual es añadido como array. En teoría cambiar esta línea por algo más performante mejorará el funcionamiento. Evitar la creación de un nuevo objeto y la transformación a array son cosas que se evaluará modificar.

Resolución:

Al principio se cambió este array por una lista, ya que cumpliría la misma función y evitaría la innecesaria creación de arrays, sin embargo no fue una solución que haya dado mejoras en el resultado final, por lo tanto, luego de investigar se determinó cambiar la lista por un HashSet, lo cual dio resultados óptimos.

Tercera medición:

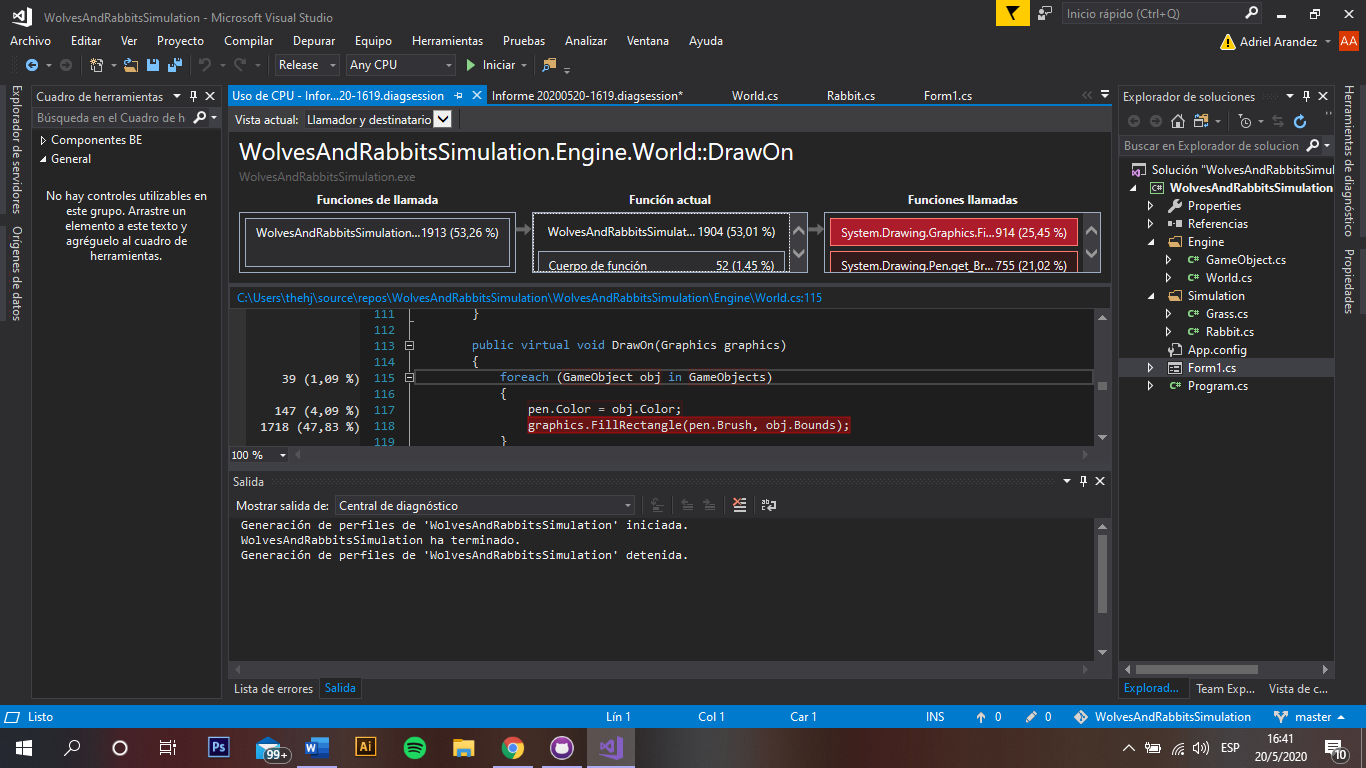


La tercera medición dio como resultado que el método DrawOn era poco performante ya que dentro del mismo se crea un Pen nuevo cada vez que es necesario graficar los objetos, por ende, si esto es reemplazado por un solo pen que se adapte al color correspondiente mejorará la ejecución.

Resolución:

Tal y como se señaló anteriormente se reemplazó la creación de pen nuevos cada vez por uno solo el cual cambia de color dependiendo del objeto a pintar.

Cuarta medición:



La cuarta medición dio como resultado que el punto más problemático seguía siendo dentro del método DrawOn, en FillRectangle, sin embargo, en base al conocimiento actual se considera que el código está lo suficientemente optimizado y se desconoce alguna otra manera en la que su funcionamiento mejore de manera significativa.

En base a estos resultados se comenzará la implementación de la clase Wolf.