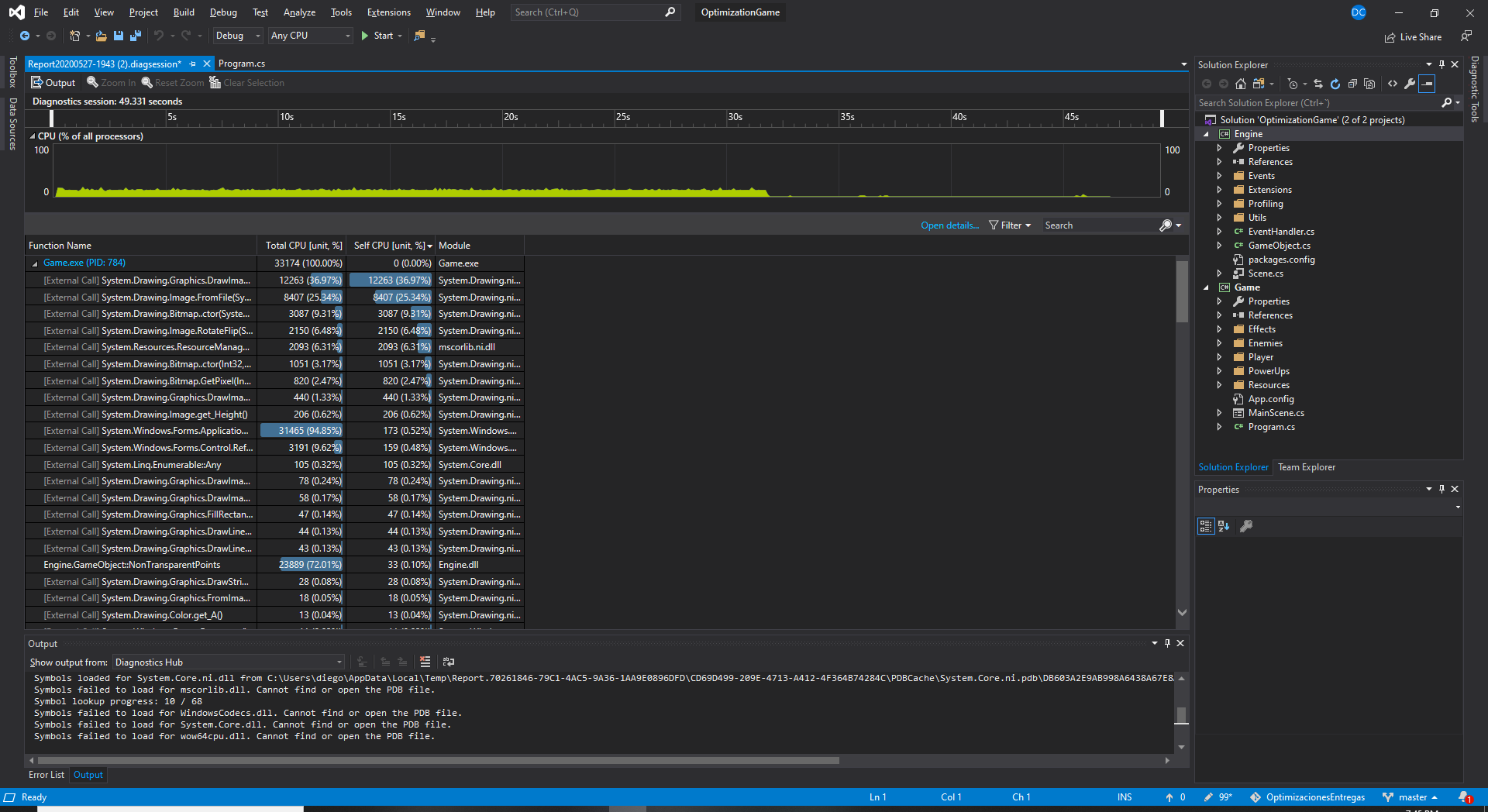
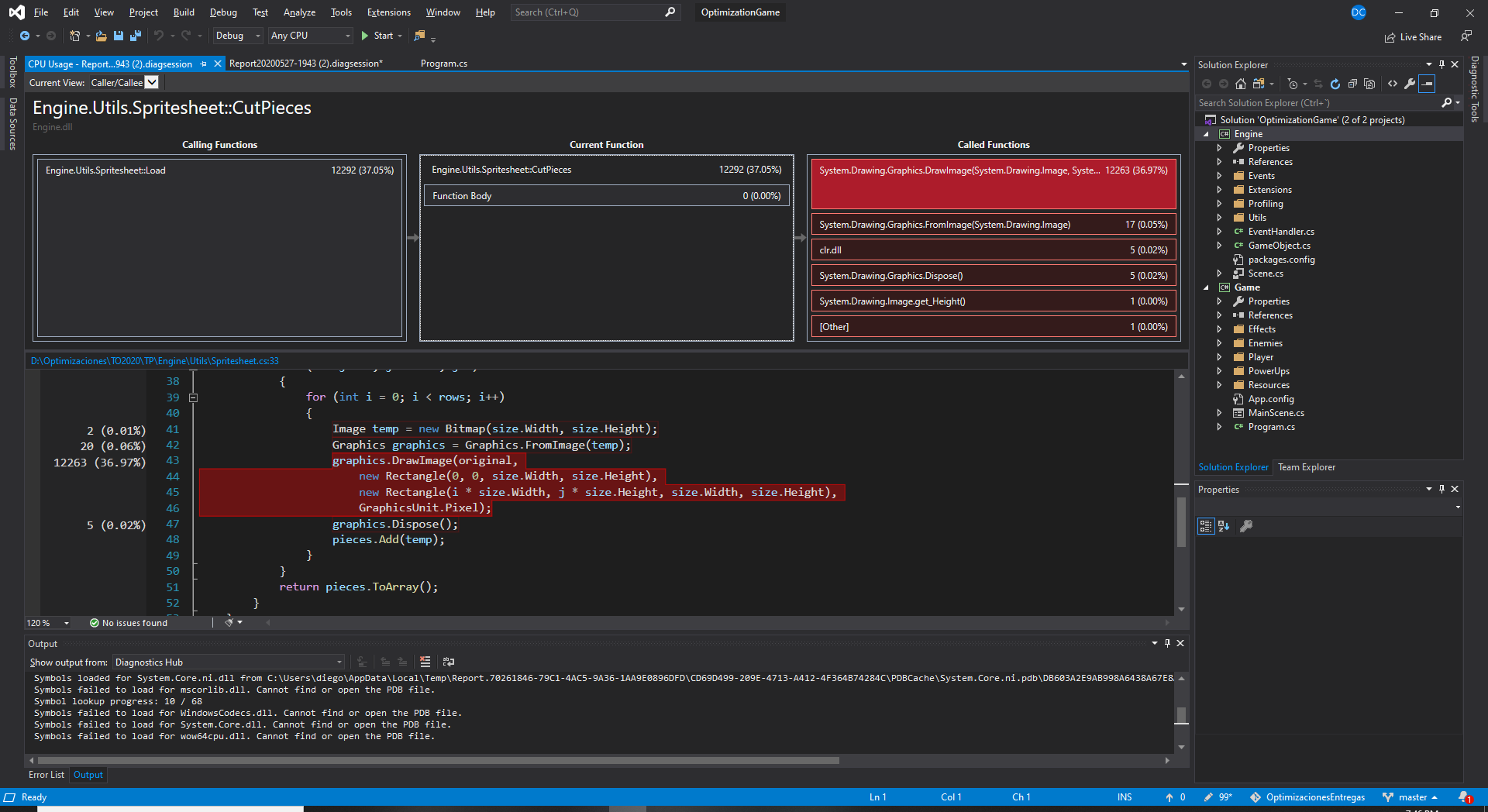
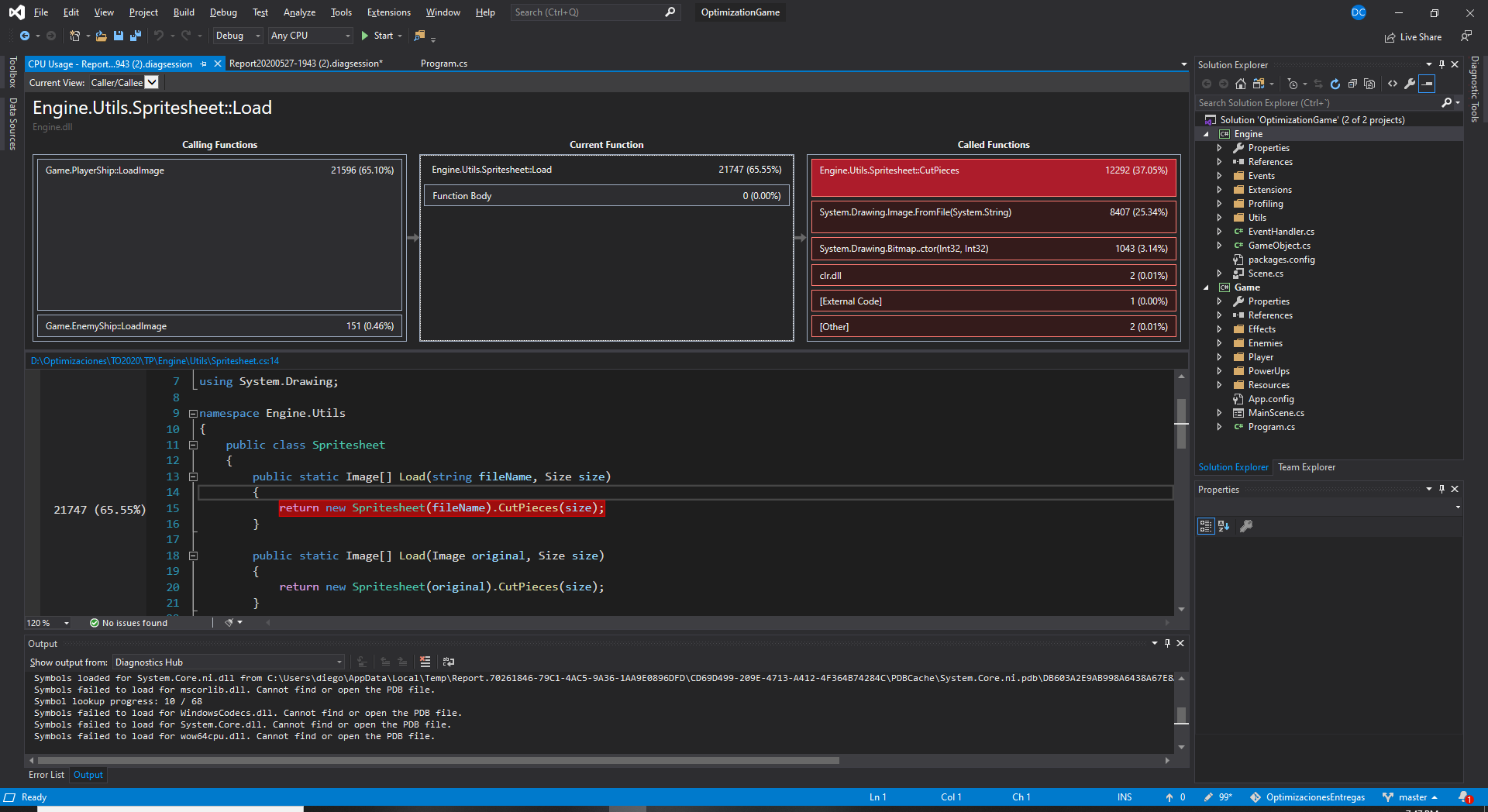
Ciclo 1:

Se realizaron las mediciones del programa, los resultados fueron los siguientes:



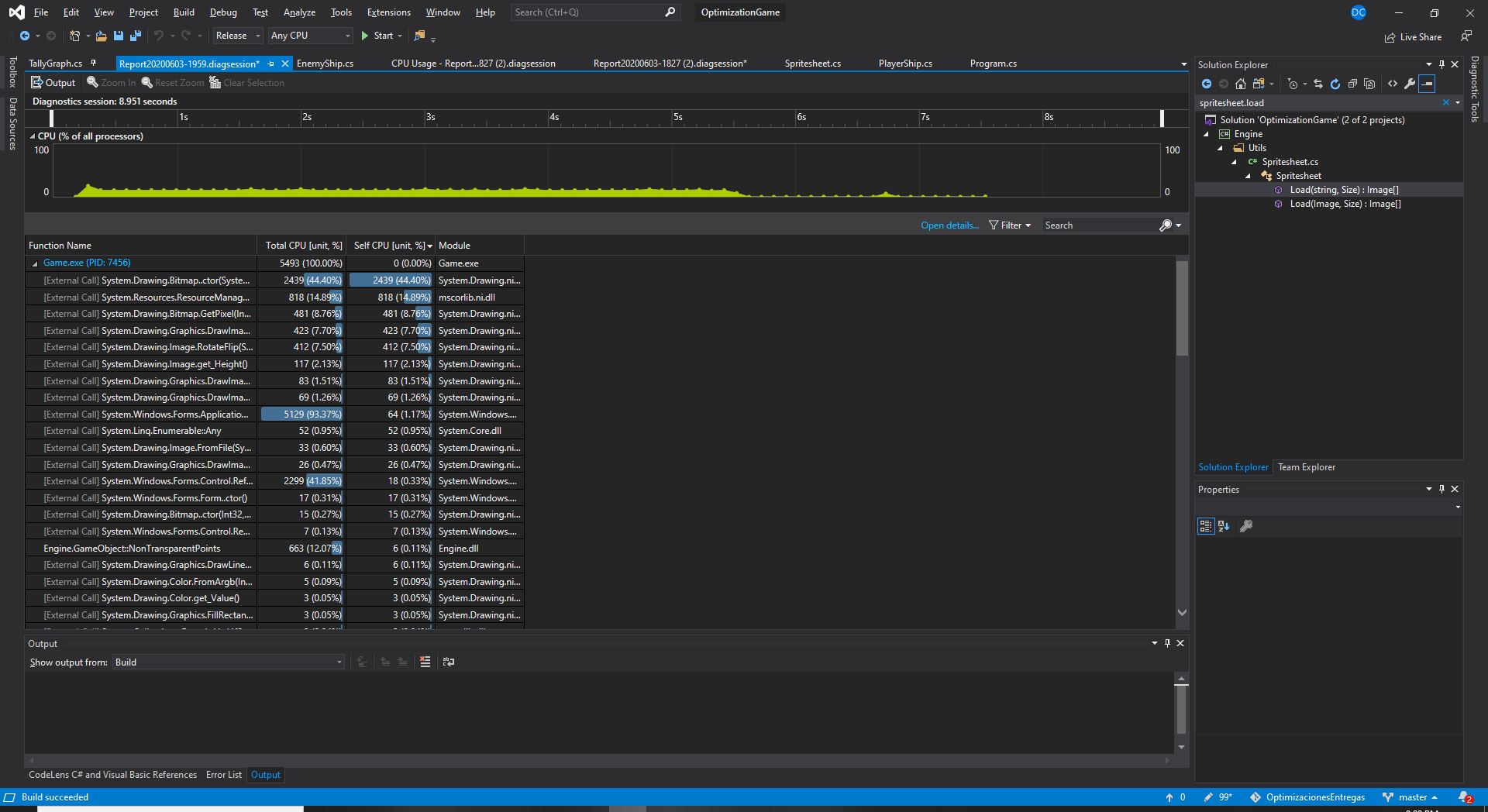




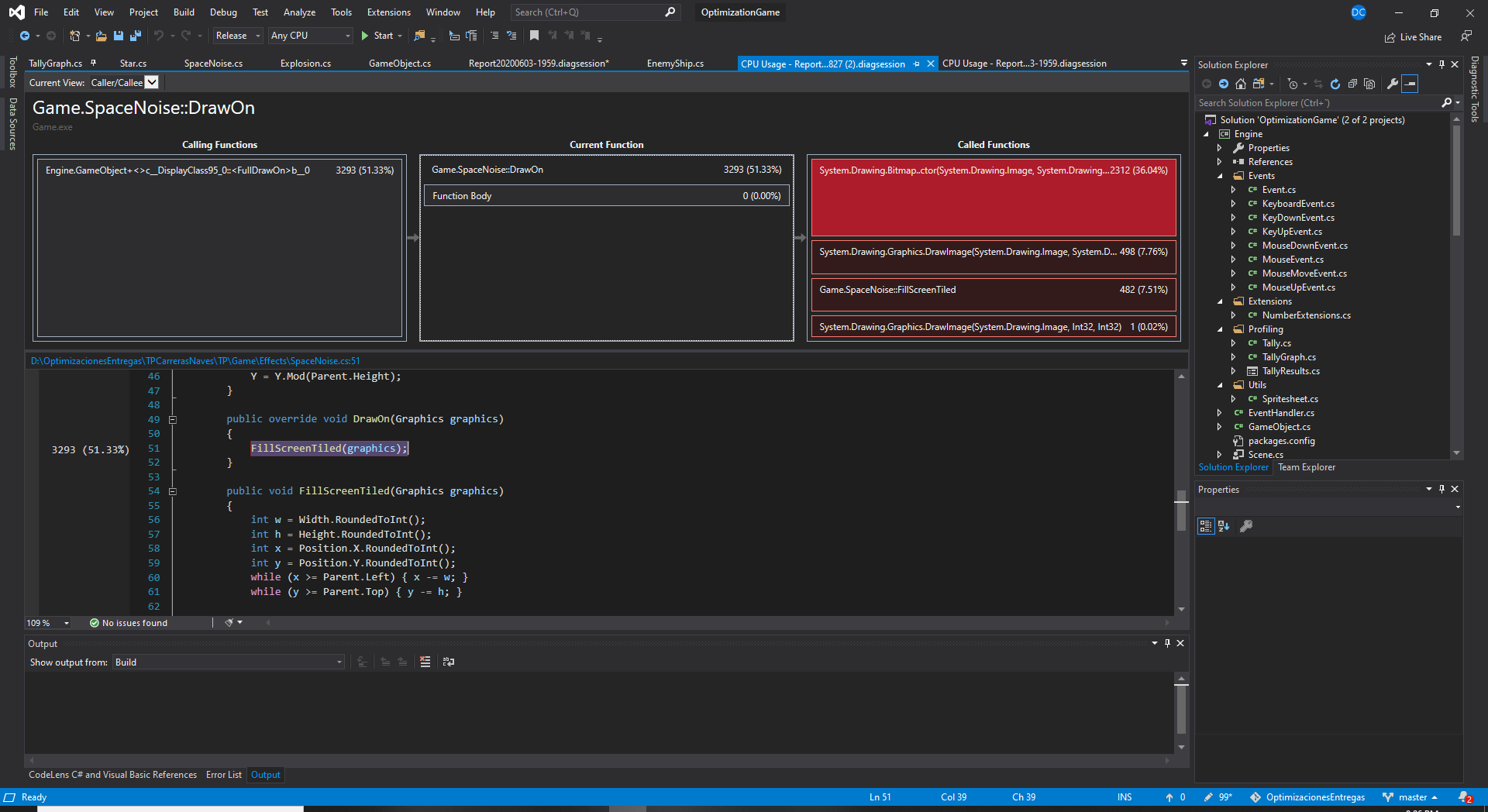
La funcion cutpieces que está causando mucha demora es referenciada por Load que luego es referenciada por el load image de enemyship y de playership. Por lo que entiendo esto carga la imagen de la nave cada vez que es llamada y la dibuja. LoadImage es llamada a su vez por drawOn. Creo que si puedo hacer que esto se llame menos veces u optimizar el código para que haga menos cálculos podre hacer que el programa corra de manera más fluida.

En la clase PlayerShip procedi a crear una variable PlayerShipImage donde se cargara la imagen y asi no tener que cargarla denuevo cada vez que DrawOn es llamado. Despues procedi a hacer lo mismo con las naves enemigas (clase EnemyShip).

El programa mejoro notablemente, paso de practicamente 0 fps por segundo a poder avanzar (se que suena como poco pero al principio no me llegaba ni al primer frame y despues llego a los 4 en unos 2 segundos). La imagen de abajo son las nuevas mediciones:



Como se puede observar el problema principal ya no es el mismo sino que ahora se encuentra en lo que a primera vista parece ser la clase SpaceNoise en la funcion DrawOn.



Intente optimizar el spaice noise ya que estaba ocasionando muchísima molestia en el código y no pude. Para no seguir trabado ahí decidí comentar las líneas y correr el profiler de nuevo para ver el siguiente problema. La única línea que modifique fue en el FullDrawOn fue DrawBoundsOn(graphics); pero no mejoro el código.

Captura de pantalla de computadora

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Como se puede observar en las siguientes imágenes, algo que está ocasionando mucha molestia en el código es el Update de la playership, específicamente en los métodos CheckForPowerUps y CheckForCollision.

Lo que esta sucediendo es que el metodo busca en todos los objetos con los que colisiona, esto incluye por ejemplo las estrellas que son muchísimas instancias. Por ende, desperdicia muchísimo tiempo y recursos de la máquina, tendría que buscar una manera en la que no haga esto.

Lo que hice fue simplemente cambiar el orden del linq para que primero identifique si el objeto es una nave enemiga y que después se fije si colisiona. Esto mejoro muchísimo la fluidez del programa, después hice exactamente lo mismo con checkforpowerups y también mejoro mucho. A continuación dejo las mediciones.

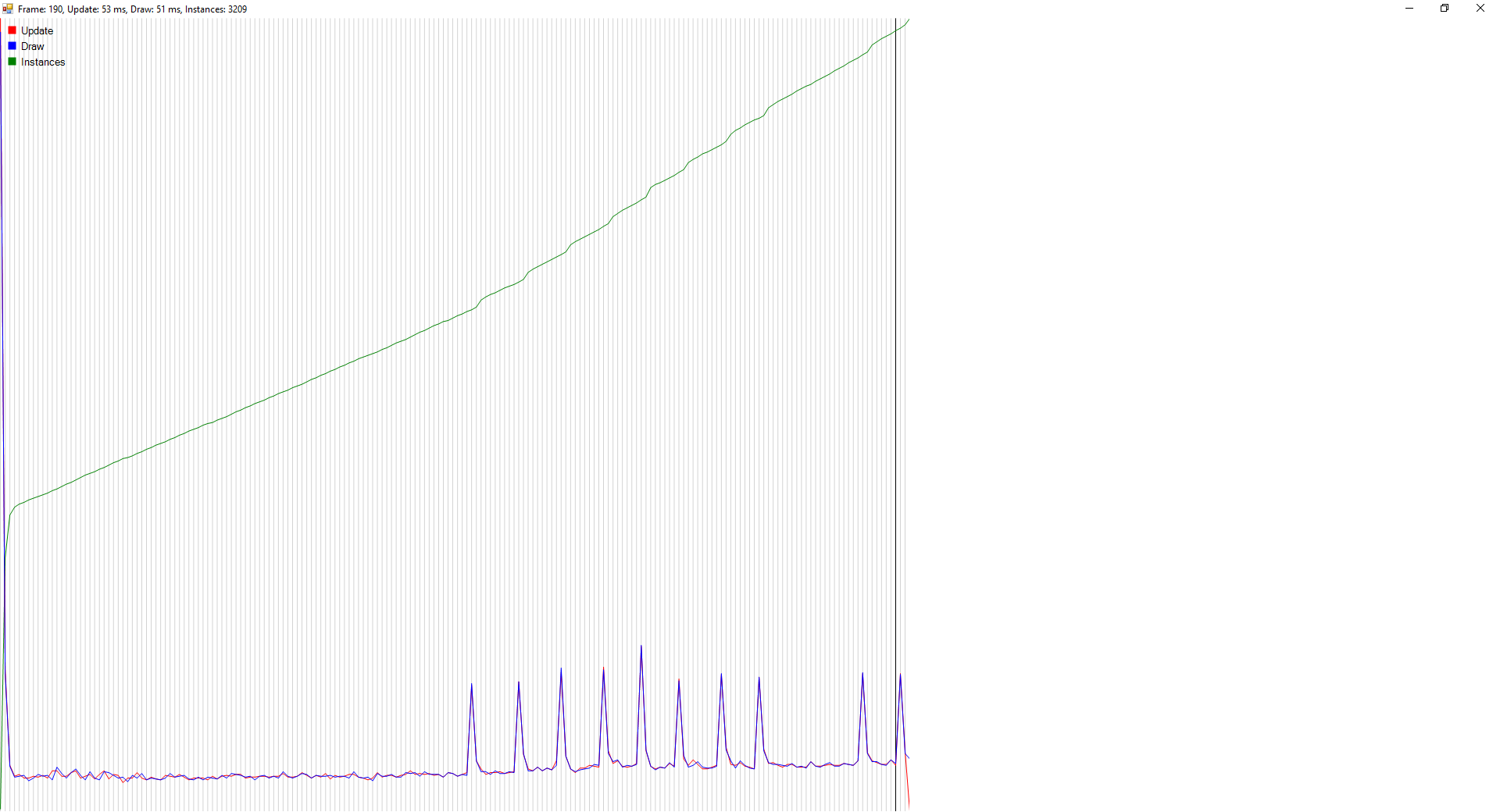
Captura de pantalla de computadora

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamente

Como se puede observar el Game.PlayerShip Update ya prácticamente no causa problemas.

A continuación pondré las siguientes mediciones:



Como se puede observar las instancias no paran de crecer de manera drástica a medida que avanza el tiempo. Esto hace que a medida que pase el tiempo en el juego este vaya cada vez más lento debido a la cantidad de instancias que hay.

Captura de pantalla de computadora

Descripción generada automáticamente

También aquí se puede observar como el FullDrawOn consume muchos recursos del programa, calculo que en parte tiene sentido ya que es el dibujado de las imágenes pero aun así intentare optimizarlo para que no consuma tanto.

Analizando más el código me di cuenta que los elementos dentro de la lista de los gameobjects es en su gran mayoría estrellas, esto se debe a que estas se instancian pero nunca se eliminan. No creo que esto afecte en el draw on de todas maneras ya que solo se dibujan los elementos visibles, de todas maneras es un problema que se debe solucionar. En cuanto al drawOn no encontré una gran manera de solucionarlo excepto por eliminar el primer DrawOn graphics que no hace nada, hasta donde puedo entender estaría dibujando 2 veces cada imagen.

Cambio 1: FullDrawOn

Elimine el primer DrawOn(graphics), pensé que esto mejoraría la velocidad del juego pero no lo hizo, después de más investigación me di cuenta de que ese drawon simplemente dibuja un fondo negro en el juego. Lo elimine de todas maneras ya que no es necesario porque ni se ve.

Cambio 2: Estrellas

La solucion que encontre fue eliminar las estrellas cuando se salen del mapa, esto seria cuando llegan su posicion x = 0. Por ende agregue el siguiente codigo:

if(this.Position.X<=0)

{

this.Delete();

}

Esto fue un éxito ya que antes las instancias subían drásticamente con el tiempo y ahora se mantienen en aproximadamente 2000 todo el tiempo.

Imagen que contiene texto

Descripción generada automáticamente

A continuación, las siguientes mediciones:

Captura de pantalla de computadora

Descripción generada automáticamente

Me di cuenta que el juego enlentece cada vez que se spawnean naves enemigas, por ende decidí indagar por ese lado. Descubrir que enemyspawner consume muchos recursos y por ende intentare ver como optimizarlo.

Lo que hice fue aplicar memoization y en vez de crear una nueva variable enemyship cree esta variable con anterioridad y luego simplemente la reutiliza. Esto mejoro levemente el código pero no mucho.

Captura de pantalla de computadora

Descripción generada automáticamente

Esto se debe ya que el problema no venía tanto por el lado de la necesidad de el memoization sino porque al buscar la imagen el juego se ralentiza.

A partir de aca me di cuenta que no estaba disparando ni haciendo mucho cuando hacia el benchmark por ende decidi comenzar a hacerlo y descubri el siguiente problema:

Captura de pantalla de computadora

Descripción generada automáticamente

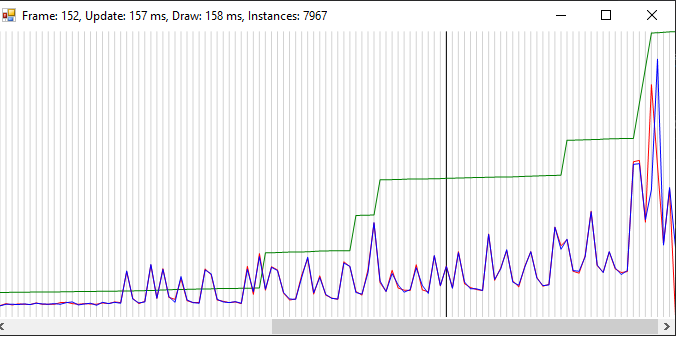
Es un problema muy parecido a uno resuelto anteriormente, la bala está buscando todos los objetos con los que colisiona y después se fija si son una nave enemiga, esto puede ser resuelto de manera muy simple al primero seleccionar las naves enemigas y después fijarse si colisionan.

Captura de pantalla de computadora

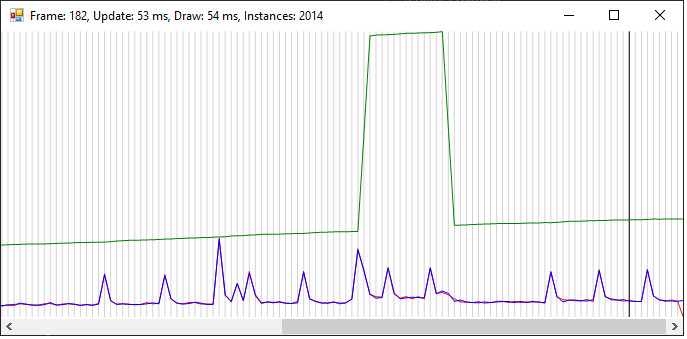
Descripción generada automáticamente

Como se puede observar la performance mejoro bastante en relación a la medición anterior.

En la siguiente medicion tambien comenze a destruir las naves enemigas, aca es cuando note que volvio mi problema de las instancias.



Esto se debe a que con cada explosión se instancian muchísimas partículas como gameobject. Lo que tendría que hacer es eliminar las instancias después de que ya no se ven. También podría comenzar a separar los objectos que no interactúan con nada de la lista de gameobjects principal, de esa manera ahorrando muchos recursos que ya no tendrán que ser recorridos.



Lo que hice fue agregar un spotwatch al update de Explosion en donde si paso 0.7 segundos desde su instancia que se destruya. Esto no afecta al gameplay en absoluto ya que la partícula no es visible por tanto tiempo. Como se puede observar esto afecto de muy buena manera la optimización del código ya que las instancias se eliminan rápidamente y no se van acumulando como hacían antes. también procedí a crearles una lista por separado para que no tengan que ser recorridos innecesariamente. A esta lista también podría agregarle las estrellas.