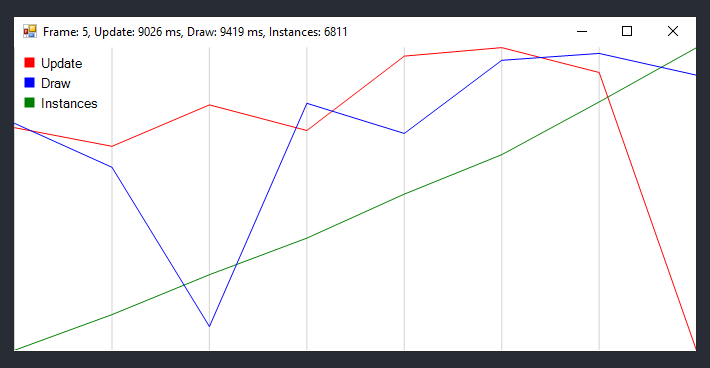
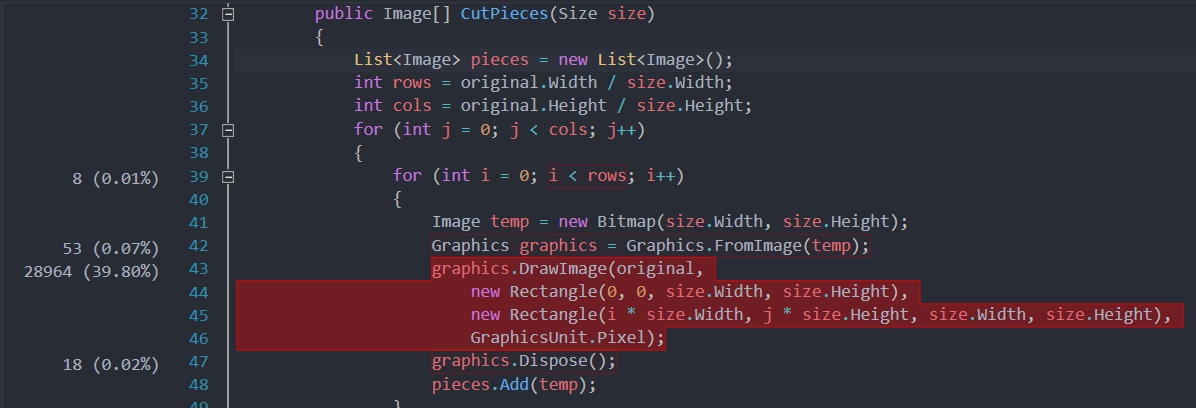
Trabajo práctico – Tecnicatura en optimización.  
 Catania, Gian Franco.

**Información del software sin optimizar.**



**-------------------------------------------------------------------------------------------------------------------------------**



En la primera imagen, se nos indica que tenemos un gran problema con las instancias ya que pareciera que hay objetos duplicados.

Tenemos también problema con el dibujado que no logra poder seguir el ritmo creando un conflicto de paralelismo.

En la segunda imagen, tenemos un método que genera cortes de un archivo de sprites.   
 Cada vez que la llaman, esta volviendo a realizar el mismo procedimiento para devolver dicha petición.

**-------------------------------------------------------------------------------------------------------------------------------**

**Hipótesis 00**:

Analizando el comportamiento, se preguntó si el problema podría ser que cada vez que se llama al método se crea una Lista y se vuelve a rellenar toda la información.   
 La solución pensada fue quitar la instancia de la lista en ese bloque de comportamiento, generar la variable como campo y creando un nuevo objeto del mismo tipo con una lista genérica.  
 Lamentablemente no surgió ninguna mejora de performance. Se está planteando que podría ser incluso peor ya que la lista nunca se limpia y se llena de imágenes recortadas ya que vuelve a agregar datos en la última posición de este.

****

**Hipótesis 01:**

Esto no mejora ningún rendimiento. Cada vez que se vuelve a pasar por la clase Spritesheet se vuelve toda la información de las variables instanciadas como campo en NULL.

Se desconoce el motivo.

**-------------------------------------------------------------------------------------------------------------------------------**

Texto

Descripción generada automáticamente

**Hipótesis 02:**

Como en la hipótesis 01 se intento evitar que se vuelva a realizar cortes. Se busco las referencias que tiene ese método:

public static Image[] Load(string fileName, Size size)

{

return new Spritesheet(fileName).CutPieces(size);

}

En la primera imagen se comenta el código de la primera línea dentro del bloque código y lo declaro campo para que se realice una sola llamada.  
  
 Esto ha dado un gran rendimiento favorable como se puede apreciar en esta imagen que verá a continuación en modo **Release**.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Como se decía, un buen rendimiento favorable, lamentablemente no se puede cerrar esta hipótesis 02 ya que esto está provocando un aspecto visualmente feo. Por el momento solo se puede apreciar que las naves están constantemente rotando.  
  
**Conclusión:**   
 Buen rendimiento, falta verificar el estado de las rotaciones de los objetos de cada corte de naves.

**-------------------------------------------------------------------------------------------------------------------------------**

**Hipótesis 03:**

**Gráfico, Gráfico de líneas

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

Analizando por encontrar la solución sobre los sprites que insistían en rotar constantemente se propuso crear una condición. Se declara una variable de tipo Booleana para indicar que una vez que se cumpla ese mini algoritmo, no vuelva repetir.

Se lograr solucionarlo y se puede decir que ha mejorado otro rendimiento favorable para el software.

**-------------------------------------------------------------------------------------------------------------------------------  
-------------------------------------------------------------------------------------------------------------------------------**