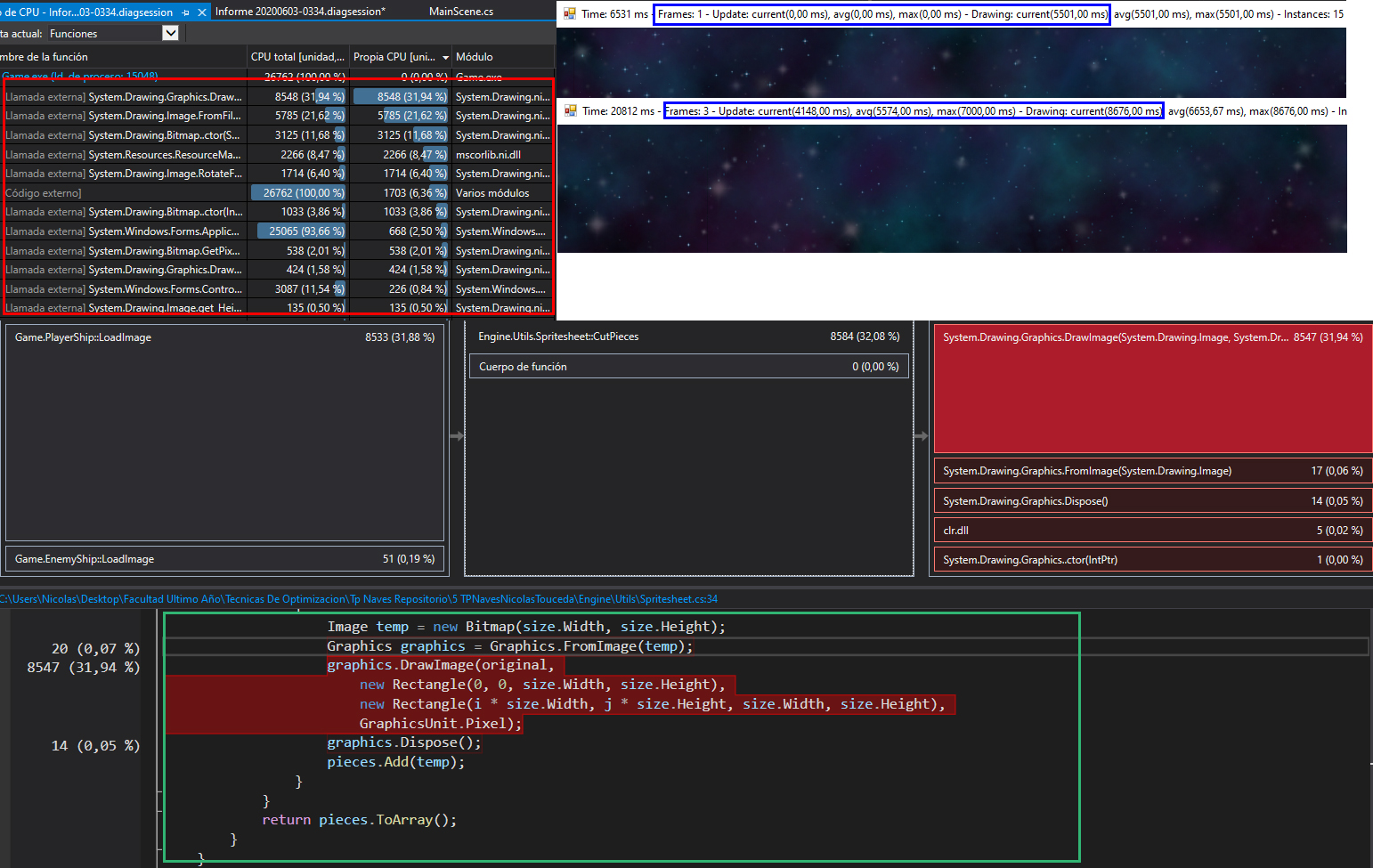
TP NAVES HOLA

Nicolas Touceda

HIPOTESIS 1

Mediciones:

Hipotesis:

Mis principales problemas gracias a las mediciones son:

Cuadro Rojo de la imagen: Encontramos el Primer HotSpot, donde el método de dibujo (System.Drawing.Graphis.Draw…) Utiliza muchos recursos y ocupa el 32% del procesador, Esto también genera mucho retraso en el programa.

Cuadro Verde de la imagen: Encontramos un bloque de código (graphics.DrawImage() que se encuneta en el método public Image[] CutPieces(Size size) manda a dibujar la lista de imágenes), donde ocupa el 32% del tiempo de la CPU.

Cuadros Azules de la imagen: Vemos que en cada Frame que se ejecuta el juego el Drawing tarda 5 a 8 segundos en dibujar. Esto es un problema porque no deja avanzar el programa.

Teniendo estas 3 mediciones en cuenta sabemos que este método draw es un problema.

Revisando el código llegue a la conclusión de que tanto el PlayerShip como EnemyShips, al momento de ser dibujadas, llaman 2 veces al método LoadImage. Lo que este método hace es desencadenar una serie de pasos, en el que, se recortan y dibujan todas las imágenes y son almacenadas en un array , finalmente este array es retornado, para ser dibujado (Esto ocurre en en CutPieces() dentro de la clase Spritesheet).

El problema de esto es que, cada ves que quiero cargar una imagen de una nave, el programa manda a recortar y crear todas las imágenes de vuelta.

Solución Propuesta:

Modificar 2 métodos para un mejor funcionamiento del programa, primero el método CutPieces voy a implementar un IF para que cree las imágenes una sola vez y no tenga que crearlas cada vez que se le llama el método LoadImage()

Voy a modificar el Método LoadImage() para acceder a mi imagen directamente por el índice de array y voy a guardar esta imagen en una variable.

Resolución:

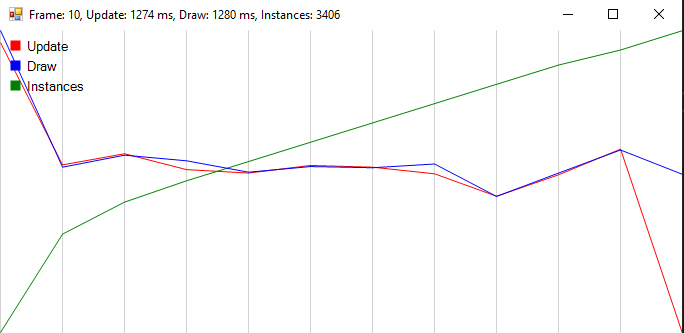


Comparando esta imagen con la de medición, llegue a la conclusión que el programa mejoro mucho, ya que el tiempo de dibujado paso de 5-8 segundos a 1-2 segundos y por ende el programa avanza mucho mas rápido, en el mismo tiempo de prueba avanzo 9 frames en vez de 3.

La Implementación y los cambios Fueron Positivos.

Hipotesis 2

Mediciones:





Hipotesis:

Como podemos ver, en la imagen de medición, hay un problema en la cantidad de instancias, estas se multiplican excesivamente frame a frame.

Revisando la lista de objetos (All Children) note que frame a frame se añaden más muchas estrellas que nunca se liberan de la memoria.

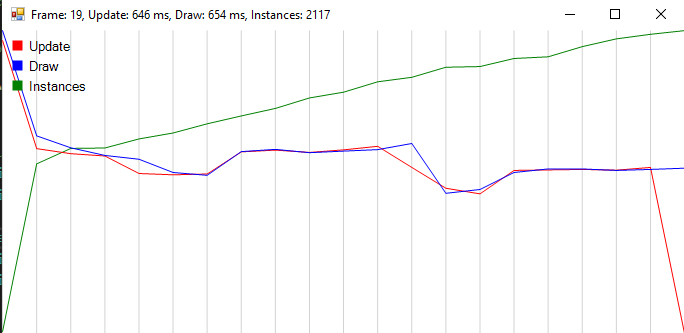
Solución Propuesta:

Lo que propongo es crear una nueva estructura de datos para separar las estrellas de AlChildren.

Una vez que tenga esto, voy a implementar algún método que elimine las estrellas que estén fuera de pantalla, de esta manera voy a bajar mucho el número de instancias.

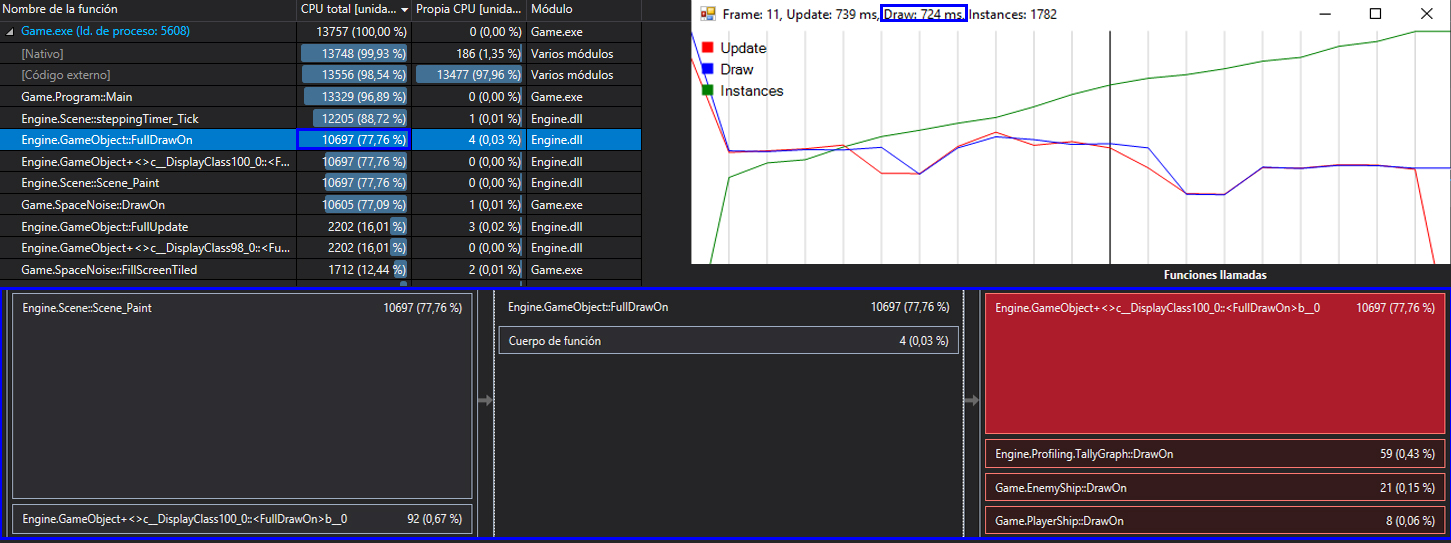
Resolucion:

Bajo mucho el numero de instancias a comparacion con la primer medicion, las estrellas se liberan al salir de la pantalla el cambio fue positivo.



Hipotesis 3

Mediciones:

Hipotesis:

Revisando las nuevas imágenes de medición, destaco un problema en el método FullDrawOn(), que ocupa un 77,77% del CPU Total, Este es llamado por la Scene\_Paint.

Entre frame y frame se tarda entre 600 a 800 ms en dibujar los objetos en pantalla

Revisando el código pude notar que el método FullDrawOn, agarra la lista de objetos, y por cada objeto se vuelve a llamar a si misma varias veces, esto provoca que el mismo objeto se dibuje más de una vez, y cuantos más objetos tenga la lista , más veces se van a dibujar.

Solución Propuesta

Propongo modificar el método FullDrawOn para que cada objeto se dibuje una sola vez.

Resolucion: