Teljesítményjellemzők vizsgálata

Szoftverfejlesztés laboratórium 2

Mérési útmutató

Készítette: Hajdu Ákos, Barta Patrik, Vörös András

Verzió: 3.1

2022.

Budapesti Műszaki és Gazdaságtudományi Egyetem

Méréstechnika és Információs Rendszerek Tanszék

# Előkészületek

## Általános tudnivalók

A mérési feladatokat és az elvégzésükhöz szükséges, a kiadott segédlet által nem tartalmazott információkat jelen útmutató tartalmazza. A feladatok dokumentálását jelen útmutató kijelölt helyeire végezzék! A feladatoknál mindig jelöljük, hogy kell-e dokumentálni, és ha igen, akkor milyen formában (táblázat, diagram, képernyőkép, forráskód, magyarázat).

**Fontos:** A feladatok megoldása során az eredmények értékelése is a feladat része: az eredmény olyan-e, amit vártunk, reális-e az az eredmény, stb. Ezeknek a végiggondolása a mérés része, szándékosan hiányoznak a „Most azt kellene látni, hogy…” részek!

## Szoftverkörnyezet

A mérés elvégzéséhez a következő szoftverelemek állnak rendelkezésre a virtuális gépen. A gépen minden szükséges fájl megtalálható a **C:\ \_profiling** mappában!

A mérés elvégzéséhez szükséges szoftverek:

* Microsoft Visual Studio 2019 Community
* Windows Performance Recorder (WPR)
* Windows Performance Analyzer (WPA)
* Microsoft Office (Word / Excel)

### Bemeneti állományok

A bemeneti állományok a C:\\_profiling mappában találhatóak. Ha a feladat nem írja elő másképp, akkor ezekből lehet dolgozni.

* \_src: a mérés során használt alkalmazások forráskódja (Visual Studio projektek)
* 1.5.1 Eredmények: előre felvett profiling eredmények
* Disk IO: előre felvett diszk IO adatok
* Teszt képek: az alkalmazás futtatása során felhasználható teszt képek

# Profiling

## Ismerkedés az alkalmazással

**Indítsa el a DemoApp.Profiling alkalmazást és próbálja ki a működését a bemeneti állományokból válogatva (Teszt képek mappa)! Nézzen bele az alkalmazás forráskódjába (beleértve a DemoApp.Filters projektet is)! Az effektek működését nem szükséges végignézni.**

*Megjegyzés: A mintaalkalmazás csak oktatási célokat szolgál, valós környezetben a hosszú ideig tartó műveleteket nem érdemes a grafikus felülethez tartozó szálon futtatni.*

Javasolt időráfordítás: maximum 5 perc.

**A feladatot nem szükséges dokumentálni.**

## Mintavételezett profiling

A következő feladatok során mintavételezett profilingot kell végrehajtani és az eredményét értékelni. A feladatok során célszerű „Release” módban futtatni a programot, mert úgy gyorsabb.

***1.2.1. Lépéssorozat futtatása mintavételezett profilinggal***

**Indítson el egy mintavételezett profilingot és hajtsa végre a következő lépéseket:**

**- Nyissa meg valamelyik képet.**

**- Alkalmazza rá az olajfestmény effektet.**

**- Zárja be az alkalmazást.**

Javasolt időráfordítás: 5 perc.

**A feladatot nem szükséges dokumentálni.**

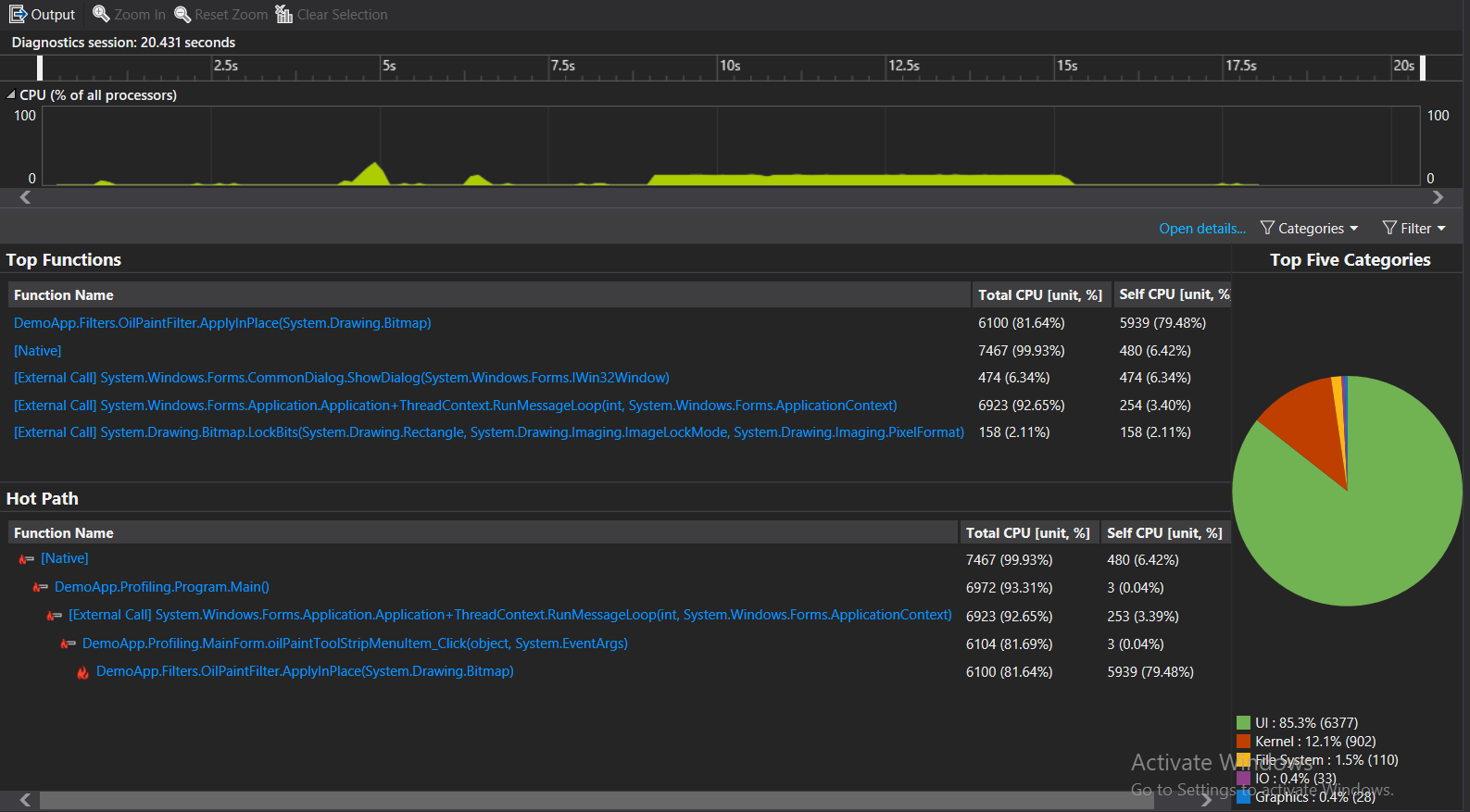
***1.2.2. Ismerkedés az eredménnyel***

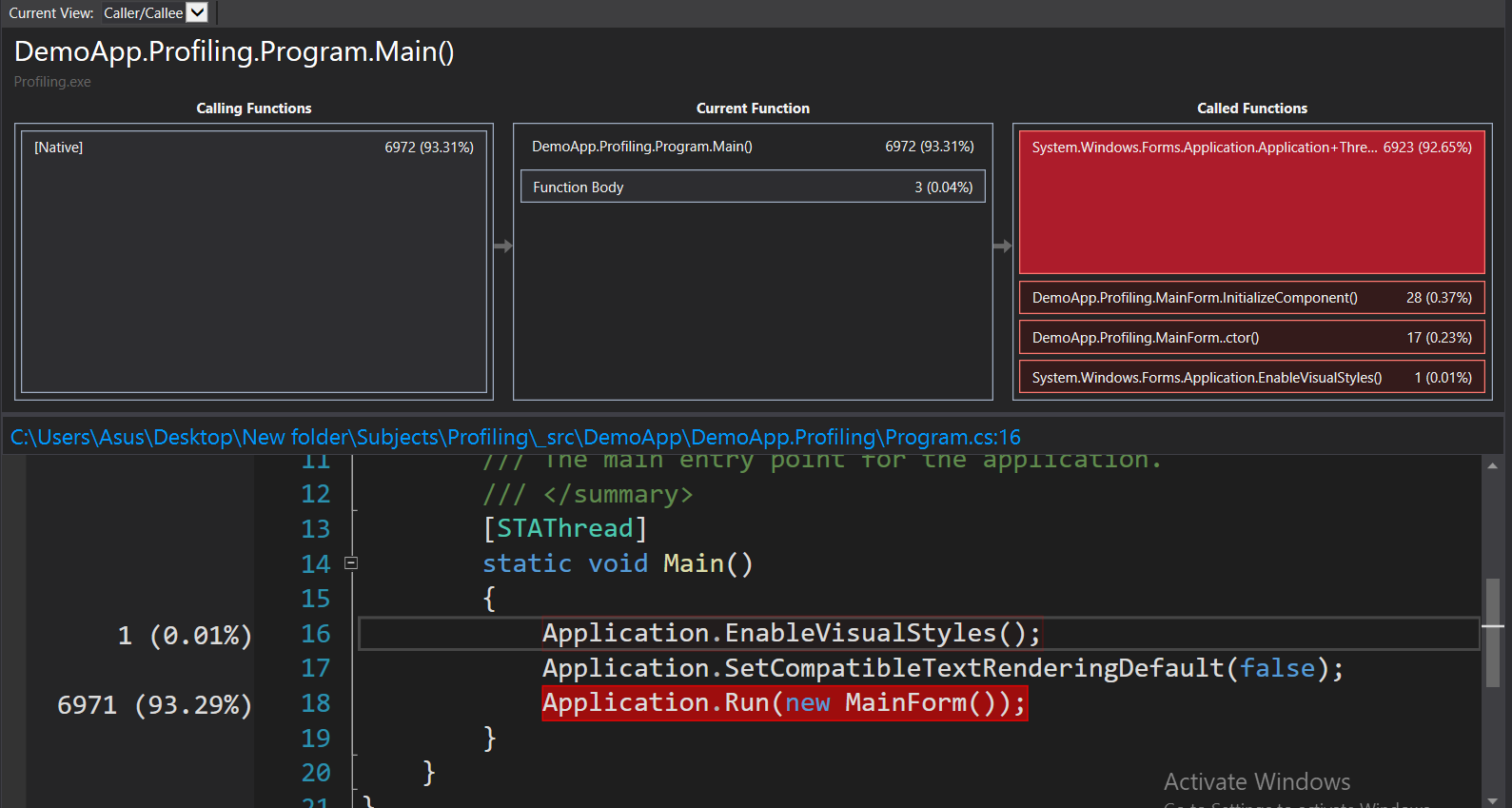
**Ismerkedjen meg a mérési eredményeket megjelenítő nézetekkel! Az áttekintő nézetben próbálja ki a “Show External Code” opciót!**

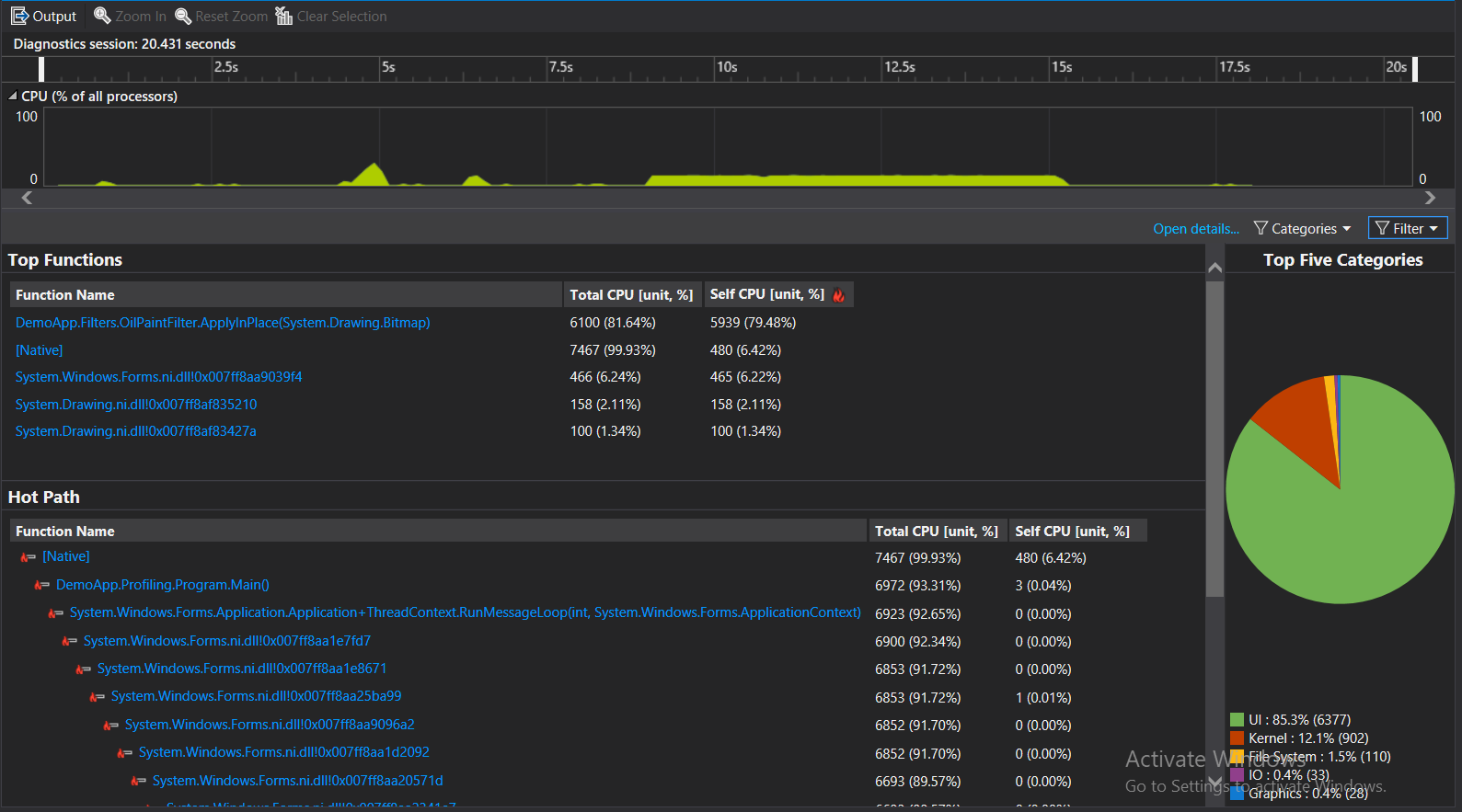
Javasolt időráfordítás: 10 perc.

**A feladat dokumentálandó (képernyőképek).**

Megoldás:







***1.2.3. Hot Path***

**Írja le röviden a Hot Path definícióját! Jelen esetben mi a Hot Path?**

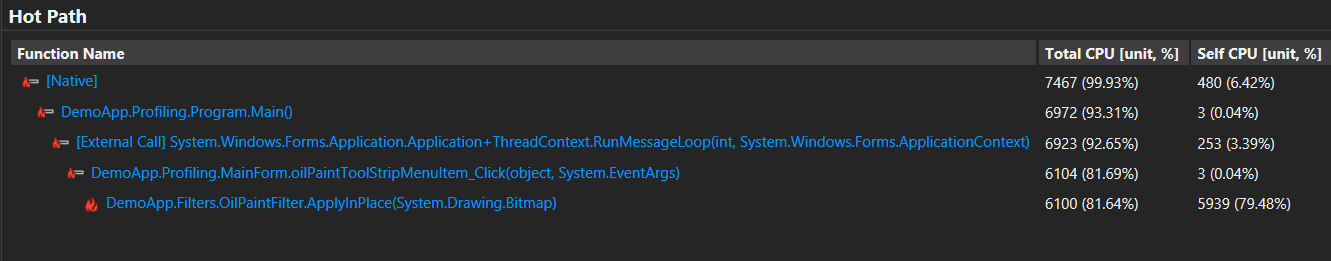
Javasolt időráfordítás: 5 perc.

**A feladat dokumentálandó (képernyőképek, magyarázat).**

Megoldás:

A hot path a call tree azon ága, ami a legtöbb CPU használatot igénylő függvényhívásokból áll. Jelen esetben ez a következőket tartalmazza:

* A WinForms alkalmazás belépési pontja
* Message loop indítása
* A filter gomb megnyomása
* A filter alkalmazása a képre (legköltségesebb)



## Instrumentált profiling

A következő feladatok során instrumentált profilingot kell végrehajtani és az eredményét értékelni.

***1.3.1. Lépéssorozat futtatása instrumentált profilinggal***

**Indítson el egy instrumentált profilingot és hajtsa végre a következő lépéseket:**

**- Nyissa meg valamelyik képet.**

**- Alkalmazza rá az olajfestmény effektet.**

**- Zárja be az alkalmazást.**

Javasolt időráfordítás: 5 perc.

**A feladatot nem szükséges dokumentálni.**

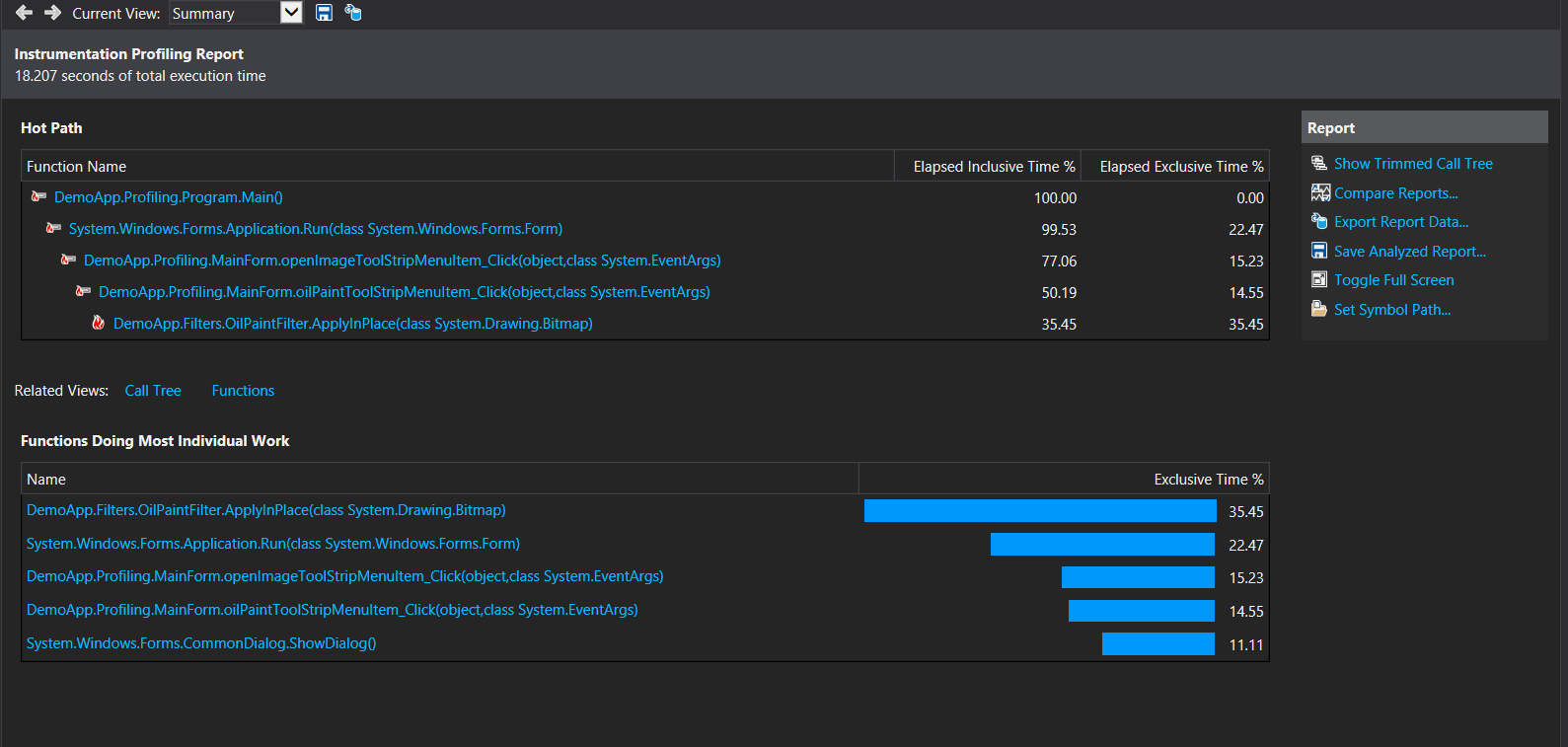
***1.3.2. Ismerkedés az eredménnyel***

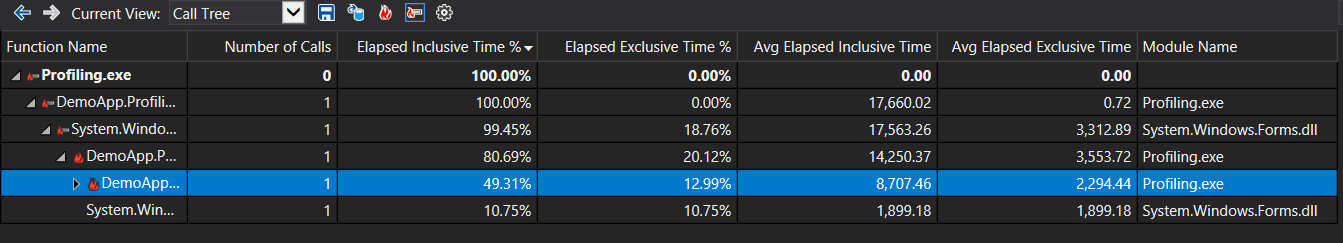
**Tekintse át a mérési eredményeket! Milyen különbségek figyelhetőek meg a mintavételezett és az instrumentált nézetek között? Ugyanaz a mért adatok mértékegysége?**

Javasolt időráfordítás: 10 perc.

**A feladat dokumentálandó (képernyőképek, magyarázat).**

Megoldás:





Az első feltűnő különbség a 2 nézet között az, hogy az instrumention report nézete nem tartalmaz grafikonokat. Ennél azonban fontosabb különbség, hogy a sampling report azt mutatja meg, hogy a profiling session alatt a különböző függvények hány mintában fordultak elő, az instrumentation report-ban pedig az látható, hogy a session teljes időtartamának mekkora részét teszi ki egy-egy függvény végrehajtása. Előbbi esetben tehát a mérőszámok egységben, utóbbi esetben pedig másodpercben értendők.

## Memória problémák keresése

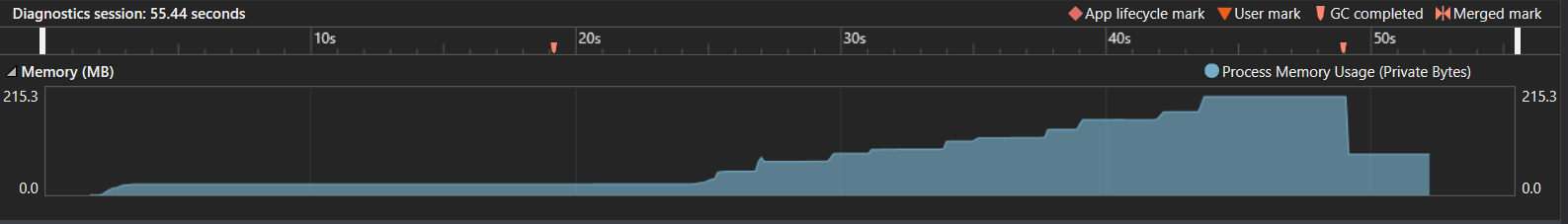
***1.4.1. Memória problémák keresése***

**A DemoApp.Profiling alkalmazásban szándékosan elrejtettünk egy hibát, amely a memóriát nem optimálisan használja ki. Indítsa el a Visual Studio „Memory Usage” eszközét erre a projektre. Nyisson meg egymás után több képet (legalább kettőt-hármat) és figyelje a memóriafogyasztást. Hajtson végre egy szemétgyűjtést (Force GC) miután megnyitott több képet. Mivel magyarázható a jelenség? *Segítség: Mi okozhat felügyelt környezetben ilyen, memóriaszivárgáshoz hasonló problémát?***

Javasolt időráfordítás: 10 perc.

**A feladat dokumentálandó (képernyőképek, magyarázat).**

Megoldás:



A profiling session során a képek megnyitásakor folyamatosan nő a memóriahasználat, ezt mutatja a grafikon középső, emelkedő szakasza. A GC kikényszerítése után a memóriahasználat hirtelen visszaesik, ez látható a grafikon utolsó szakaszán. A problémát az okozza, hogy a Bitmap egy wrapper class a gdiplus graphics library felett, ami unmanaged. Emiatt a Bitmap instance-eket vagy a GC-nek kell felszabadítania, vagy manuálisan Dispose()-t kell hívni rajtuk, mivel a Bitmap class implementálja az IDisposable interface-t. A profiling során a képek megnyitásakor nem futott le a GC, így azok a memóriában maradtak, amíg explicit nem hívtuk meg a GC-t.

***1.4.2. Memória probléma megszüntetése (nem kötelező feladat, IMSc pontért megoldható)***

**Keresse meg a memória pazarlás pontos okát és javítsa ki! Futtassa le újra a „Memory Usage” eszközt, hogy meggyőződjön a javítás sikereségéről!**

Javasolt időráfordítás: 10 perc.

**A feladat dokumentálandó (képernyőképek, forráskód részlet, magyarázat).**

Megoldás:

Mivel a Bitmap az Image class leszármazottja, ami implementálja az IDisposable interface-t, a kiválasztott képen Dispose()-t kell hívni. Az openImageToolStripMenuItem\_Click() így módosul:

if (ofd.ShowDialog() == DialogResult.OK)

{

pictureBox.Image?.Dispose();

try

{

pictureBox.Image = new Bitmap(ofd.FileName);

}

catch (Exception ex)

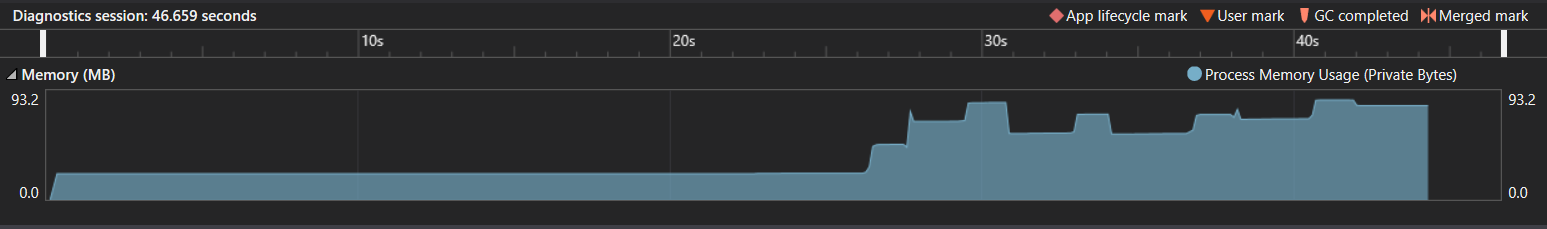
{

MessageBox.Show("Error opening image: " + ex.Message);

}

}

A ?. operátorra azért van szükség, mert az első kép megnyitásakor a pictureBox.Image null-t ad vissza, ilyenkor még nincs mit dispose-olni. A Dispose() segítségével a felesleges memóriahasználatot megszüntettük.



## Mérési adatok elemzése

***1.5.1. Adatok áttekintése***

**Egy kép különböző méretű változatain előre elvégeztünk méréseket (1.5.1 Eredmények mappa). A CSV fájlban megtalálhatóak a képek paraméterei (fájlnév, szélesség, magasság, pixelszám, fájlméret) illetve az olajfestmény effekt általunk mért futási ideje.**

**Tekintse át a képeket és a mérési eredményeket!**

Javasolt időráfordítás: maximum 5 perc.

**A feladatot nem szükséges dokumentálni.**

***1.5.2. Futási idő a pixelszám függvényében***

**Ábrázolja az effekt futási idejét a pixelszám függvényében! *Ügyeljen a megfelelő diagramtípus meg­vá­lasz­tá­sára!* Milyen függvénnyel írható le a pixelszám és a futási idő közötti összefüggés?**

Javasolt időráfordítás: 15 perc.

**A feladat dokumentálandó (diagram, magyarázat).**

Megoldás:

A pixelszám és a futási idő közötti összefüggést lineáris közelítéssel a **d = 0.0009 p – 67.882** függvény írja le, ahol d a futási időt (ms), p pedig a pixelek számát jelöli.

***1.5.3. Futási idő a szélesség/magasság függvényében***

**Ábrázolja az effekt futási idejét a kép szélességének vagy magasságának függvényében. Mi okozza a kiugró értéket? Milyen függvénnyel írható le a szélesség/magasság és a futási idő közötti összefüggés?**

Javasolt időráfordítás: 10 perc.

**A feladat dokumentálandó (diagram, magyarázat).**

Megoldás:

A szélesség és a futási idő közötti összefüggést lineáris közelítéssel a **d = 1.9941 w –1513.7** függvény írja le, ahol d a futási időt (ms), w pedig a szélességet (px) jelöli.

A megyer\_1007.jpg képnél a kiugró értéket az okozza, hogy a szélessége megegyezik a megyer\_2460.jpg kép szélességével, de a magassága annak kb. 41%-a, emiatt az algoritmusnak kevesebb soron kell végigmennie.

***1.5.4. Futási idő a fájlméret függvényében (nem kötelező feladat, IMSc pontért megoldható)***

**Ábrázolja az effekt futási idejét a fájlméret függvényében. Mi okozza a kiugró értéket? Milyen függvénnyel írható le a fájlméret és a futási idő közötti összefüggés?**

Javasolt időráfordítás: 10 perc.

**A feladat dokumentálandó (diagram, magyarázat).**

Megoldás:

A file méret és a futási idő közötti összefüggést lineáris közelítéssel a **d = 0.009 s + 990.32** függvény írja le, ahol d a futási időt (ms), s pedig a file méretet (bytes) jelöli.

A grafikon alapján azt várnánk, hogy a megyer\_1968.jpg esetében nagyobb futási időt kapunk, mivel a file mérete a megyer\_2460.jpg kép méretének kb. 2x-ese. Azonban az előbbi kép kevesebb pixelt tartalmaz, az algoritmus pedig a pixeleken iterál végig, így a futási idő az első képnél kisebb lesz.

# Nyomkövetés

A nyomkövetéshez nincs szükség Visual Studio-ra, ezért a teljesítmény javítása érdekében érdemes a programot lefordítani (Release módban), majd Visual Studio-t bezárva a projekt „bin/Release” mappájából futtatni.

## Nyomkövetés rögzítése

***2.1.1. Rögzítés***

**Indítson nyomkövetést a WPR segítségével, minden beépített “Resource Analysis” profilt bekapcsolva! A tárolt nyomot állományba rögzítse (Logging mode)! A nyomkövetés során az alábbi lépéseket hajtsa végre:**

**Indítsa el a DemoApp.Profiling alkalmazást!**

**- Nyisson meg egy képet!**

**- Alkalmazza rá az olajfestmény effektet!**

**- Nyisson meg egy másik képet!**

**- Alkalmazza rá az olajfestmény effektet!**

Javasolt időráfordítás: 5 perc.

**A feladatot nem szükséges dokumentálni.**

## Eredmények megtekintése WPA-ban

***2.2.1. Ismerkedés a WPA-val***

**Indítsa el a Windows Performance Analyzer eszközt és töltse be az előző lépésben rögzített nyomot!   
Tekintse át a felület képességeit!**

Javasolt maximális időráfordítás: 10 perc.

**A feladatot nem szükséges dokumentálni.**

***2.2.2. Viselkedés felderítése, CPU***

**Konfiguráljon fel CPU Usage (Sampled) nézetet, amely az alkalmazás CPU használatát mutatja grafikusan!**

**Konfigurálja be úgy a nézetet, hogy a CPU használat táblázatos megjelenítése is látható legyen!**

**A nézetet állítsa be úgy, hogy az alkalmazás által használt modulokat és függvényeket is mutassa!**

**Állítson be szűrőt, hogy csak a releváns (az általunk mért alkalmazáshoz tartozó) adatok látszódjanak!**

**Grafikusan azonosítsa az effekt meghívását a teljes CPU használatban kijelöléssel!**

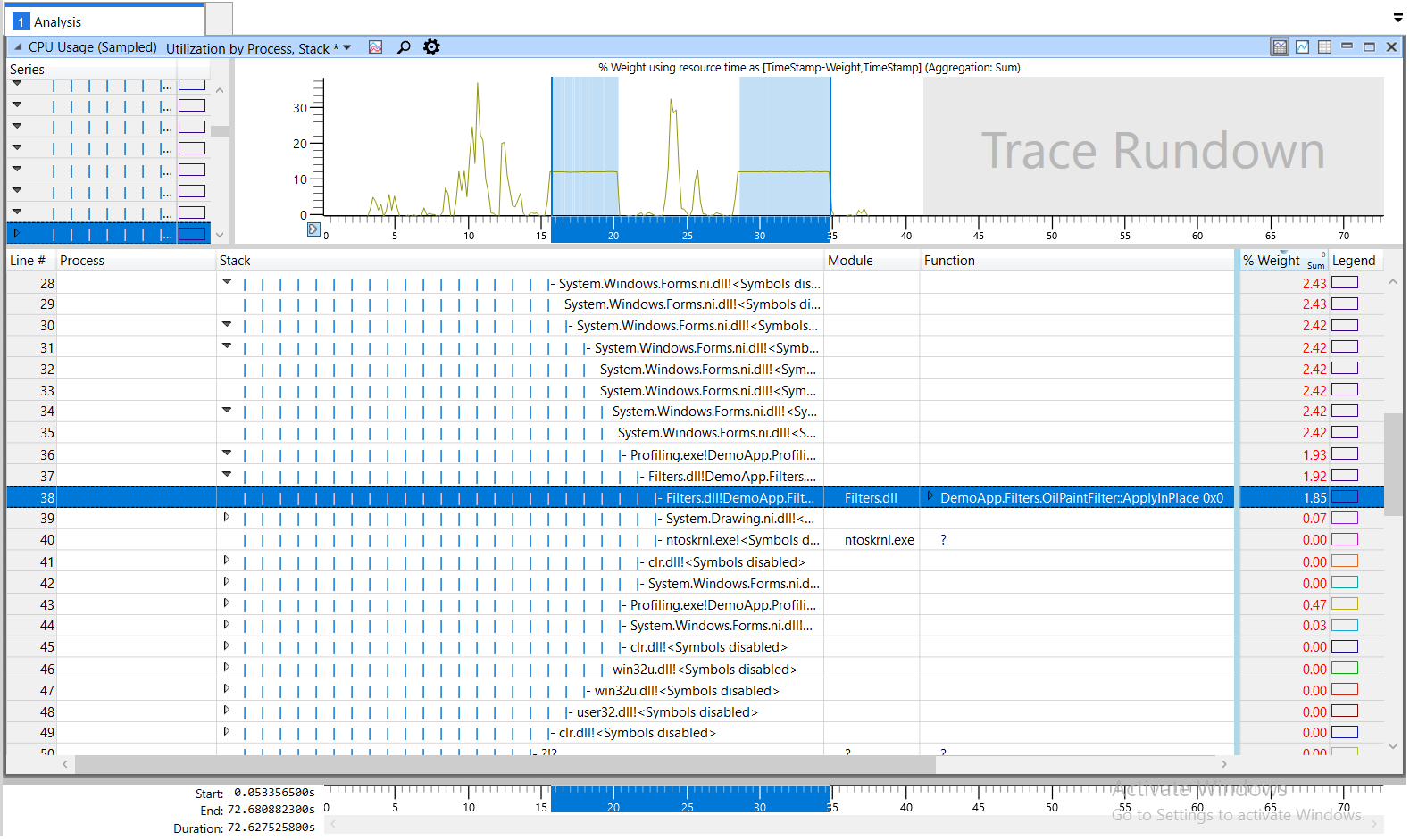
**Képernyőképpel támassza alá, amin a táblázatos nézetben látszik a kijelölt függvény (oldja fel a szimbólumokat), illetve ennek hatására a grafikonon a CPU használat! A szimbólumok feloldása előtt érdemes beállítani, hogy csak a mi alkalmazásunk könyvtárában keressen (lásd a segédletet).**

***Fontos: A feladat lényege és a megoldás értékelésének fő szempontja, hogy minél átláthatóbb és egyértelműbb nézet álljon elő. Igyekezzen minden fölösleges részlet kiszűrésére és a releváns adatok megfelelő prezentálására. A segédletben szerepelnek tippek a nézetek konfigurálására és szűrésére.***

Javasolt időráfordítás: 20 perc.

**A feladat dokumentálandó (képernyőképek, magyarázat).**

Megoldás:

  
  
A kijelölt függvény a DemoApp.Filters project OilPaintFilter file-ban lévő ApplyInPlace() metódusa, ami a filter effect-ért felelős. A táblázat Stack oszlopában a korábban a VS-ben látott expanded hot path-hoz hasonló nézet látható, a Module oszlop pedig azt mutatja meg, hogy ez a függvény a DemoApp.Filters project DLL-jében található.

***2.2.3. Viselkedés felderítése, diszk és fájl I/O***

**A virtuális gépen megtalálható egy előre felvett nyom (Disk IO könyvtárban). Azt rögzítettük, ahogy a FiltersDemo.exe nevű alkalmazással megnyitottunk néhány képet. A feladat kideríteni a diszk és fájl I/O nézetek segítségével, hogy melyik képet hányszor nyitottuk meg.**

**Konfiguráljon fel egy nézetet, amely a rendszer diszkhasználatát (Disk Usage) és fájlműveleteit (File I/O) mutatja grafikusan! Konfigurálja be úgy a nézeteket, hogy a táblázatos megjelenítések is láthatóak legyenek!**

**Állítsa be, hogy mindkét nézetben láthatóak legyenek az alkalmazás által használt állományok is! (A “Disk Usage” nézet szerkesztőjében ehhez a “Path Name” attribútumot kell hozzáadni, a “File I/O” nézetben a “File Name” jelöli a fájl nevét.)**

**Mutassa meg a File I/O nézetben, hogy mely képeket olvasta be az alkalmazás (Read esemény)!**

**Mely fájloknál nyúlt az alkalmazás a merevlemezhez (Disk I/O nézet)?**

**Ugyanazt mutatja a két grafikon? Mivel magyarázható a jelenség?**

**Az egyik képet kétszer nyitottuk meg. A fájl I/O időket vizsgálva mit tapasztalunk? Mivel magyarázható a jelenség?**

***Fontos: A feladat lényege és a megoldás értékelésének fő szempontja, hogy minél átláthatóbb és egyértelműbb nézet álljon elő. Igyekezzen minden fölösleges részlet kiszűrésére és a releváns adatok megfelelő prezentálására. A segédletben szerepelnek tippek a nézetek konfigurálására és szűrésére.***

Javasolt időráfordítás: 25 perc.

**A feladat dokumentálandó (képernyőképek, magyarázat).**

Megoldás:



A File I/O nézetben az [Event Type]:~="Read" [Process]:~="FiltersDemo.exe" [File Name]:~=".jpg" filter-t alkalmazva látható, hogy az alkalmazás a C:\inputs\steve2\steve\_w2500.jpg és C:\inputs\steve2\steve\_w1000.jpg képeket olvasta be.

A Disk Usage nézetben a [Process]:~="FiltersDemo.exe" filter-t alkalmazva látható, hogy az alkalmazás a képeknél, valamint az ieframe.dll file-nál nyúlt a disk-hez.

A chart-ok között 2 különbséget figyelhetünk meg. Az alsó ábrán nem látszik az ieframe.dll-hez tartozó File I/O művelet, de ez csak azért van, mert beállítottuk a [File Name]:~=".jpg" filter-t, hogy csak a képek látszódjanak az alsó chart-on. A filter-t kivéve már a dll-hez tartozó spike is megjelenne rajta. A másik különbség, hogy az alsó chart-on látható 3. spike-hoz nincs hozzátartozó spike a felső chart-on. Ennek az az oka, hogy a steve\_w2500.jpg megnyitása után a file bekerült a disk cache-be, így a következő megnyitáskor (3. spike az alsó chart-on) már nem volt szükség egy újabb disk műveletre, elég volt kiolvasni a disk cache-ből a file-t.