Optimizacija i performanse U Unity igri

Optimizacija igre:

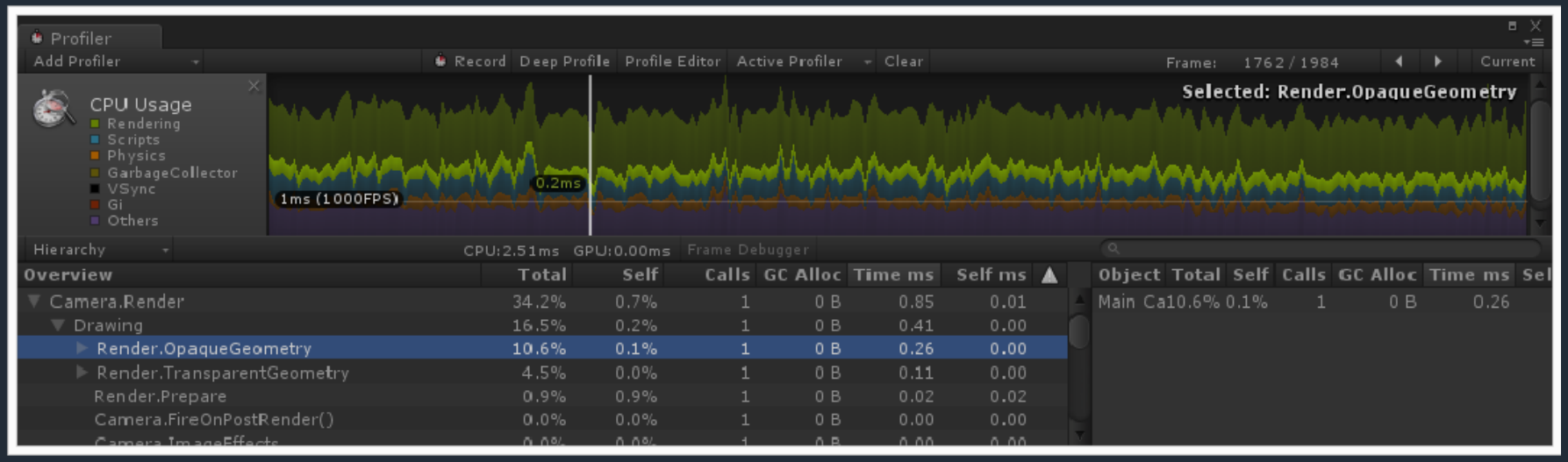
- FPS:
  - CPU
    - Kod
    - Fizika
    - Particles
  - GPU:
    - Draw calls
    - Shaderi
    - Image effects(post processing)
- „Štucavice”
  - Bugovi u frameratu
  - GC.Collection
  - Build
  - AI
  - Loading
- Memoriy
  - Održavanje male memorije na uređaju
  - Rupe u kodu i igri

UNITY PROFILER

- CPU:
  - Hijararhija i Timeline
- GPU:
  - Frame debugger
- Audio
- Physics
- Memory
- Rendering

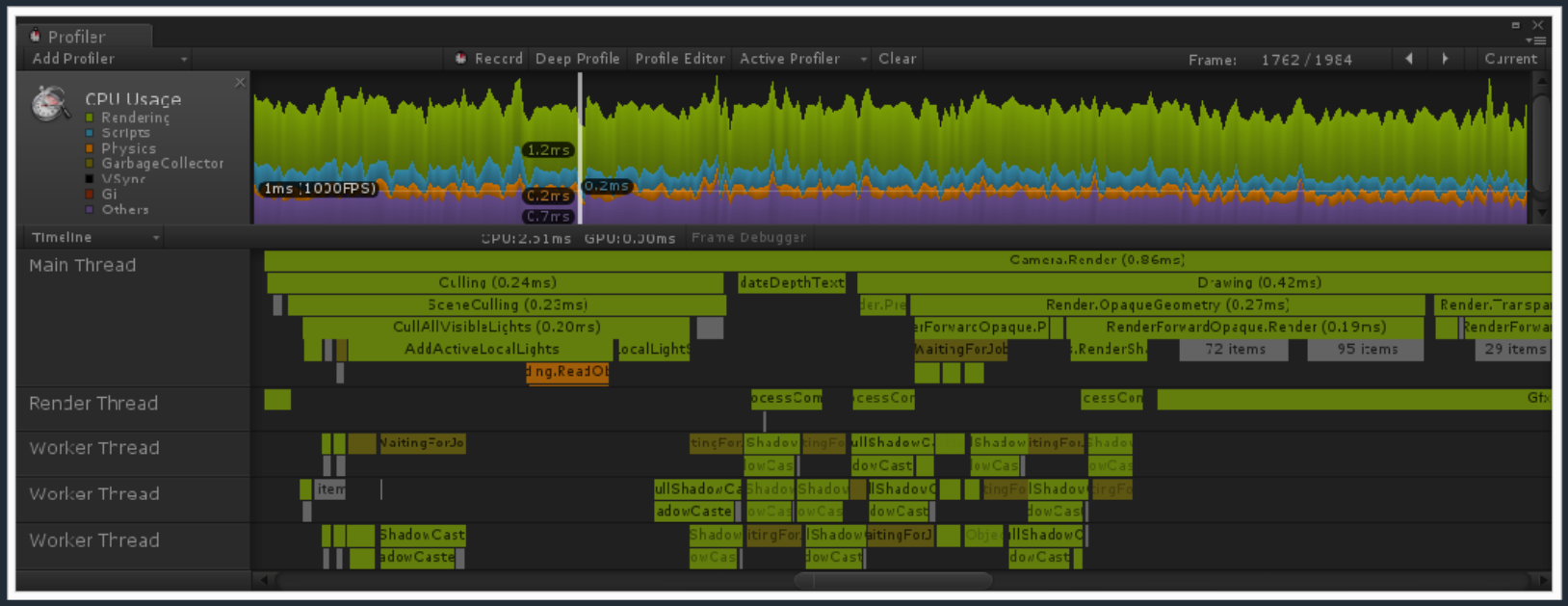
CPU PROFILER

- Iskorištenost CPU kroz vrijeme za metode
- Memory potrošnja
- Remote profiling za točni uređaj
- Deep profile (SAMO U EDITORU)
  - Jedino za male scene

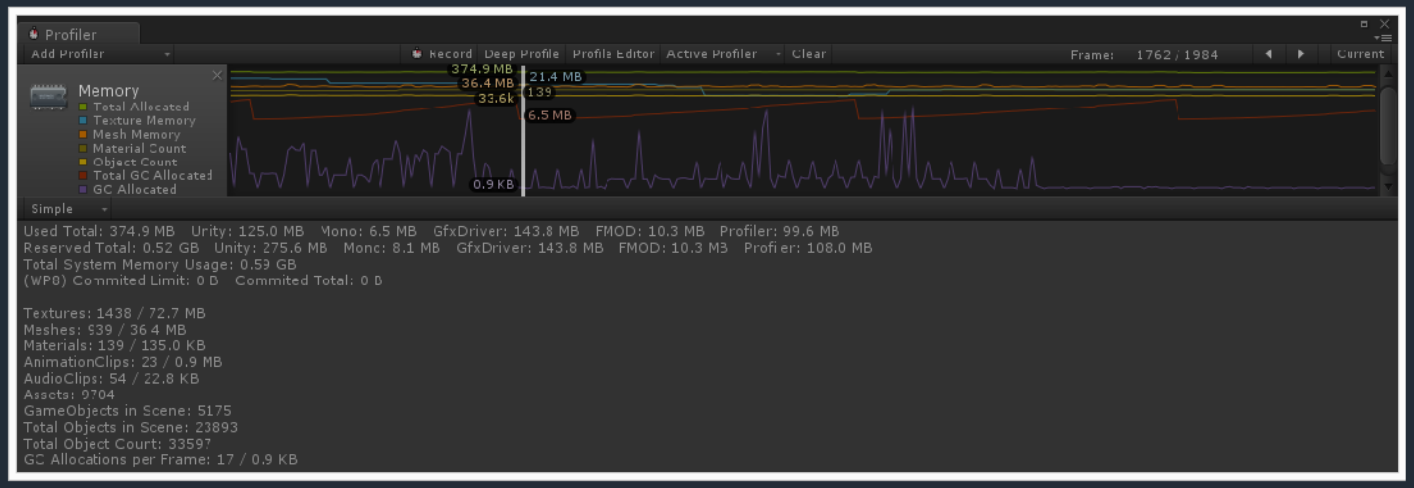


CPU PROFILER – TIMELINE

- Odličan za pregled višedretnih zadataka i metoda
- Provedba kada se koja scripta zove

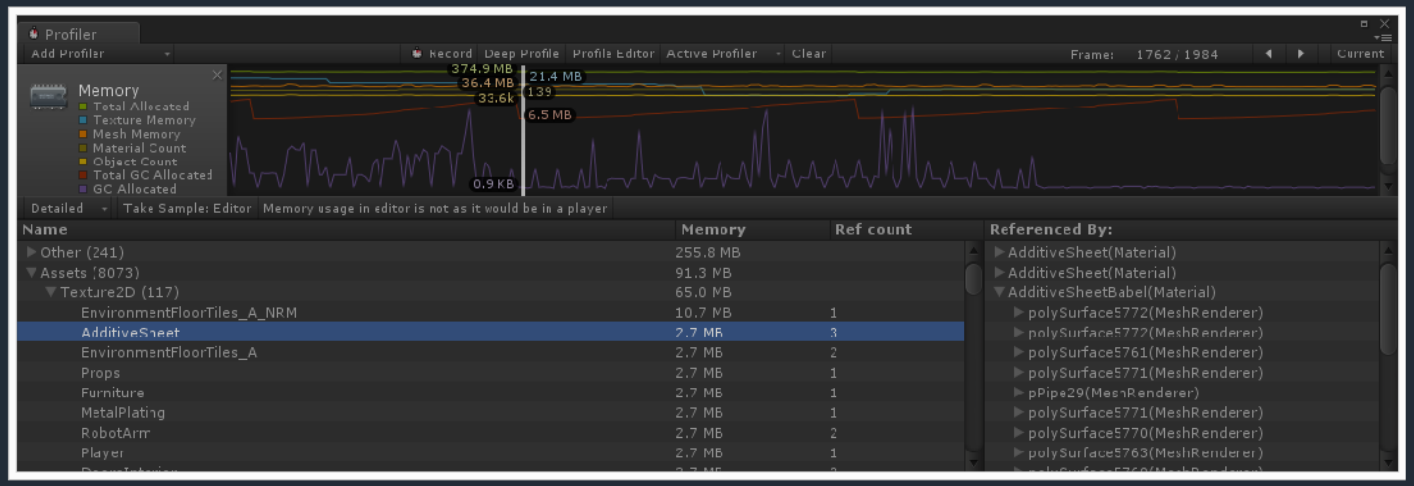


MEMORY PROFILER



- Unity sprema djelove memorije od OS-a
- Monodevelop zauzima dio memorije u RAM-u
- GfxMemory je dio memorije u VRAM-u

MEMORY PROFILER



- Detail view is taken as a snapshot (too expensive for per frame)
- Reference view See where an object is referenced – Good for leak hunting

LOADING.

LOADING..

LOADING...

LOADING DATA

- Vrste podataka:
  - Scene
  - Asseti
  - Resursi
  - Streaming
  - WWW
  - Itd.
- Učitavanje preko Unitya ili custom
- Učitavanje sinkronizirano i asinkronizirano

LOADING – ASINKRONIZIRANO

- Application.LoadLevel*Async
- AssetBundle.Load*Async
- 5.0: Resources.LoadAsync
- „Batch” veličina i dalje bitna!
  - Opterečenje glavne dretve CPU-a
- 5.0: Učitavanje textura u pozadini

LOADING – MEMORY

- Učitavanje u pozadinske dretve, problem je ako prva scena nema pred učitavanje u dretvu
- Brisanje prijašnje scene iz RAM i dretvi
- Asseti se unloadaju samo par sekundi prije nove scene
- Riješenje: Empty scena prije prave scene ili lažna tranzicija
- 5.x: Više scena se učitava i kontrolira preko API-a

LOADING – SERIJALIZACIJA

- Općenito jako brzo
- Lošije ako se upgreadala verzija!!!
- .NET udara jedno na prvi objekt
  - Neke platforme koriste „serialization weaver” (WP8)
- Teško, ali dobro:
  - „Blobified dana
  - Mapiranje memorje

LOADING – ASSETBUNDLES

- Optimizacija pleyerove veličine (najčešće najzahtjevniji objekt na sceni)
- Optimiziranje brzine builda
- Stvaranje stavki memorije po skupinama dok se učitava
  - Korištenje WWW.LoadFromCacheOrDownload

UNLOADING

-5.0 Micanje scene i procesa sa glavne dretve
    - U sekundarnu dretvu
    - Brisanje
- Zadržavanje GC-a do sljedećeg GC-a
- Pametno stavljanje GC-a
    - Resources.UnloadUnusedAssets/GC.Collect
- Unloading
    - Resources.Unload

GC Performanse

- Boehm GC scenira cijeli projekt i traži reference
  - Mark and sweep
- Brisanje gabaga
- Slaganje cache-a

GC OPTIMIZATIONS

- Reduce the work GC has to do
  - Optimize code by ensuring fewer references
- Reduce the frequency for GC to run
  - Avoid creating garbage

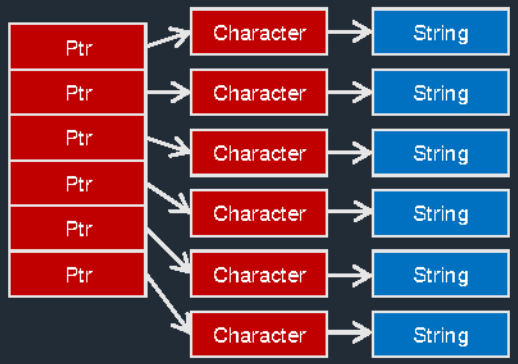
REDUCE GC WORK-a

- Pakiranje podataka direktno i sito
  - Strukture umjesto klasa
  - ECS
- Konstantna memorija iz cache-a, a manje sa HDD ili SSD-a

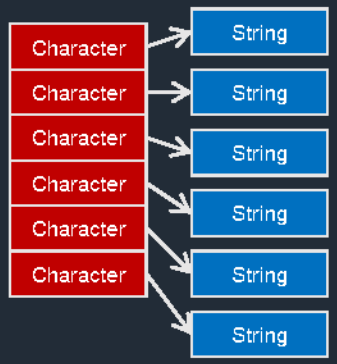
- Split dana – Spremanje vrijednosnih tipova iz indirektnih podataka
  - Float, Vector3, Int itd.
  - Strings, Klase

PRIMJER Game Code (GC) optimizacije

```
class Character
{
    public String Name;
    public Vector3 Pos;
}
Character[] characters;
```




```
struct Character
{
    public String Name;
    public Vector3 Pos;
}
Character[] characters;
```



REZULTATI

- GC za array dužine do 1.000.000 podataka na Mac Book Pro

- Klase sa nasumičnom memorijom : 35ms
- Klase sa linearnom memorijom: 20ms
- Strukture 10ms
- Strukture bez stringova: 0.18ms

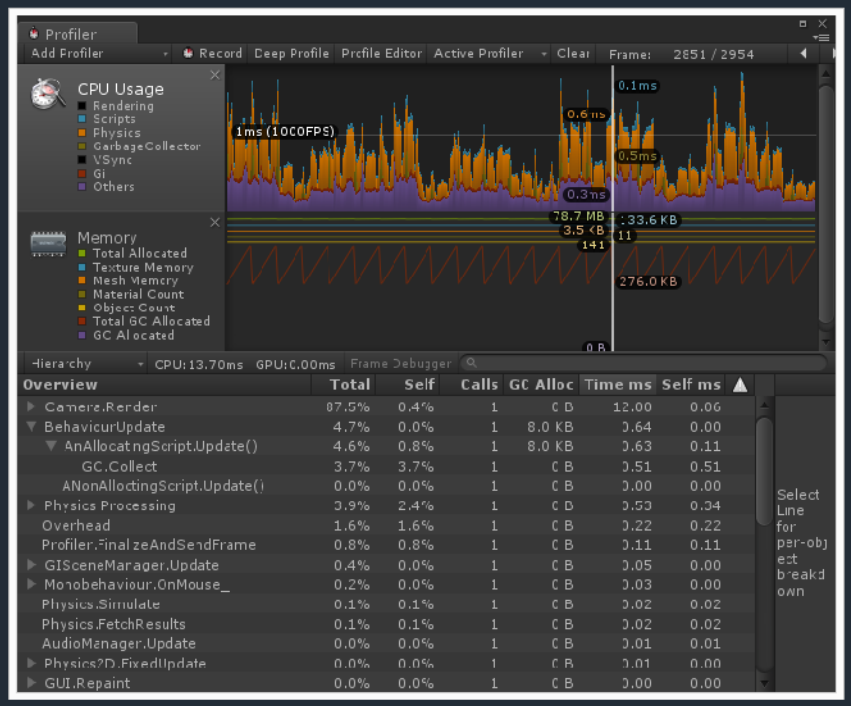
MINIMIZIRANJE ALOKACIJE

- Ponovno korištenje privremene memorije
  - Ako je data u procesuiranju I potrebni su svaki frame, alocirajte ih na buffer za reuse
- Alocirajte reusable objekte na cache
- Koristite strukture umjesto klasa
  - Strukture se pozivaju izravno
  - Klase se pozivaju sa mogućno štucavicom preko vrste (Monobehavior, RenderPipeline itd.)
- NE KORSTITE (ONGUI) – Koristite UI!

PRIMJER ALOKACIJA

```
public class AnAllocatingScript : MonoBehaviour {
    class WorkData {
        public int value;
    }
    WorkData CreateWorkData() {
        return new WorkData();
    }
    void ProcessWorkData(WorkData data) {
        int[] TempWorkBuffer = new int[2*1024];
        TempWorkBuffer[0] = data.value;
    }
    void Update () {
        ProcessWorkData(CreateWorkData());
    }
}
```

```
public class ANonAllocatingScript : MonoBehaviour {
    struct WorkData {
        public int value;
    }
    WorkData CreateWorkData() {
        return new WorkData();
    }
    int[] TempWorkBuffer = new int[2*1024];
    void ProcessWorkData(WorkData data) {
        TempWorkBuffer[0] = data.value;
    }
    void Update () {
        ProcessWorkData(CreateWorkData());
    }
}
```



POOLING

- Ciljevi:
  - Eliminira GC štucavice
  - Eliminira probleme sa Instantitate/Destroy garbagom i opterećenjem
- Problemi
  - Potrebno napraviti „zagrijavanje” odnosno ranije pozivanje prije nego što proradi
  - Povećava za 5-15% veličinu builda (problem na mobilnim platformama)

POOLING – KAKO?

- Preskače strukture koje se kose sa game codom
- Staviti pooling da krene sa loadingom same igre
- Najmanje opterećujući načini stvaranja objekata:
  - SetActive()
  - .enable za specifične komponente
  - Maknuti sa ekrana
- Brisanje objekta i stvaranje najviše troši!

GENERAL PERFORMANCE

- Avoid GameObject.Find and relatives
  - Use direct linkage or managers
  - Retain GameObject and component references
- Avoid excessive managed/native transitions
- Not everything needs to be a MonoBehaviour
- Deep hierarchies are costly

NAJZAHTJEVNIJE ZA NAPRAVITI (ne i najbolje):

- Oslobađajte main thread ponekad
- Streamning podataka
- Forsiranje SIMD na svim platformama
- Spremanje hijararhije kao jednog objekta

ZAKLJUČAK

- Učitavanje asinkronizirano je brže za velike igre
- Pazite rednosljed kojim se koristi memorija
- Reusajte (ponovo koristite) objekte umjesto da ih opet stvarate za smanjene opterecenja CPU-a
- Spremajte u Cache podatke koji se stalno koriste za brži rad igre
- Korsitite profiler za pratiti stane igre i greške