Tehnici moderne de optimizare în grafica pe calculator utilizând Vulkan

Absolvent
Mihalache Sebastian-Ștefan

Coordonator științific
Prof. Dr. Stupariu Mihai-Sorin

București, iulie 2025

Motivație

1. Procesarea unui volum mare de date în real-time

2. Maximizarea performanței prin valorificarea capacităților arhitecturilor oferite de plăcile video

3. Generarea unor efecte grafice impresionante

4. Folosirea unui API grafic modern Vulkan pentru a beneficia de optimizări avansate:
   - minimizarea transferurilor CPU-GPU
   - optimizarea alocării și utilizării memoriei
   - paralelizarea întregului proces de calcul prin divizarea și distribuirea cât mai eficientă a sarcinilor computaționale pe placa video
   - pipeline grafic flexibil, ușor de personalizat

Infrastructură

1. Etapă esențială pentru o aplicație cross-platform - Windows + Linux

2. Gestionarea bibliotecilor externe

3. Automation scripts pentru setup
   - instalarea și configurarea Vulkan SDK
   - configurare git hook - post-checkout și post-merge
   - apelarea premake5 cu parametrii specifici platformei

4. Premake5
   - configurarea arhitecturii pe x64, utilizând C++20
   - crearea configurațiilor de Debug și Release
   - legăturile de compilare statică pentru bibliotecile GLFW și ImGui
   - compilează automat noile schimbări aduse unui fișier ce definește un shader

GitHub Actions

5. CI/CD Pipeline
   - validează compatibilitatea aplicației pe Windows și Linux
   - distribuția aplicației către utilizatori

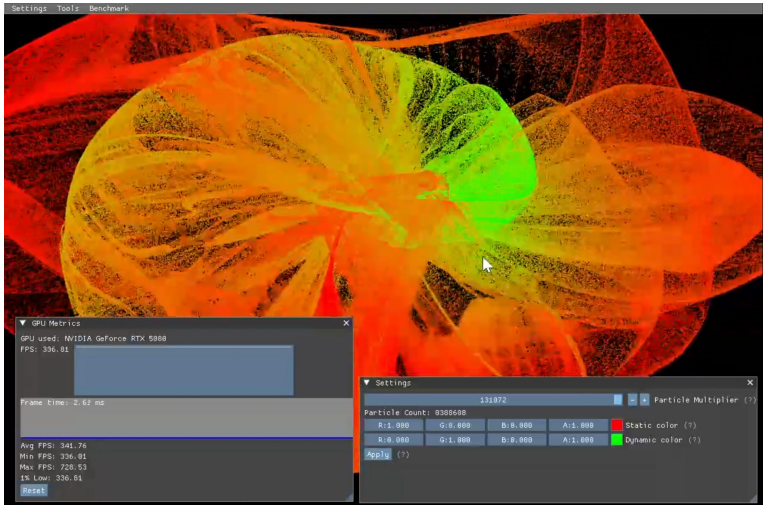
Aplicația desktop

Un simulator interactiv de particule, ce oferă o reprezentare vizuală a mișcării particulelor, influențată de setările și acțiunile utilizatorului.

Utilizatorul poate acționa asupra particulelor prin definirea unui punct de atracție pe planul 2D.

Utilizatorul poate modifica:
- numărul total de particule redate
- culoare statică = particulele în repaus
- culoare dinamică = particulele în mișcare

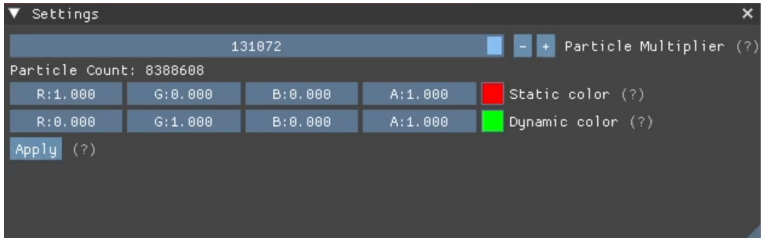
Sistem de Benchmark
- 5 teste predefinite
- posibilitatea de a capta istoricul evenimentelor generate de utilizator



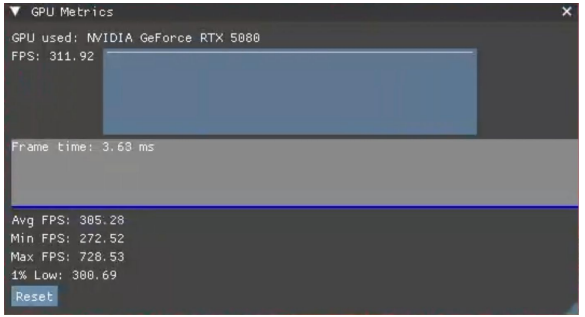
Interfața grafică

- Meniul principal:
  - Settings
  - Tools
  - Benchmark

- Settings:
  - numărul total de particule redate
  - culoarea statică
  - culoarea dinamică



- GPU Metrics:
  - FPS = rata de cadre pe secundă
  - Frame Time = timpul de procesare pentru un singur cadru
  - FPS mediu, FPS minim și FPS maxim



Arhitectura

Concepte arhitecturale pentru optimizarea aplicației:

- Exploatarea pe deplin a arhitecturii paralele masive a unităților de procesare grafică
- Etapele de rendering și simulare sunt procesate integral pe GPU
- Buffers specializate pentru maximizarea performanței

Tehnologiile folosite:

- Limbajul de programare ***C++***
- API-ul grafic ***Vulkan*** - interacțiunea cu placa video
- Biblioteca ***GLFW*** (Graphics Library Framework) - gestionarea ferestrei, evenimentelor de intrare
- Biblioteca ***GLM***(OpenGL Mathematics) - structuri și funcții matematice esențiale
- Biblioteca ***ImGui*** - instrumente pentru intefața vizuală a aplicației
- Biblioteca ***nlohmann*** - o abordare modernă pentru procesarea fișierelor JSON

Buffers

```
struct UniformBufferObject
{
    glm::mat4 projection;
    glm::vec4 staticColor;
    glm::vec4 dynamicColor;
};
```

- partajarea datelor actualizate foarte rar între CPU și GPU
- păstrează matricea de proiecție și cele 2 culori din meniul de setări

- mecanism extrem de rapid de a transmite date de dimensiuni mici
- valori inserate direct în pipeline-ul grafic
- transmise prin comenzile de rendering

```
struct PushConstants
{
    uint32_t enabled;
    float timestep;
    glm::vec2 attractor;
};
```

```
struct Particle
{
    glm::vec2 position;
    glm::vec2 velocity;
};
```

- Shader Storage Buffer Object
- gestionarea volumelor mari de date exclusiv pe GPU
- stocarea directă a datelor în VRAM

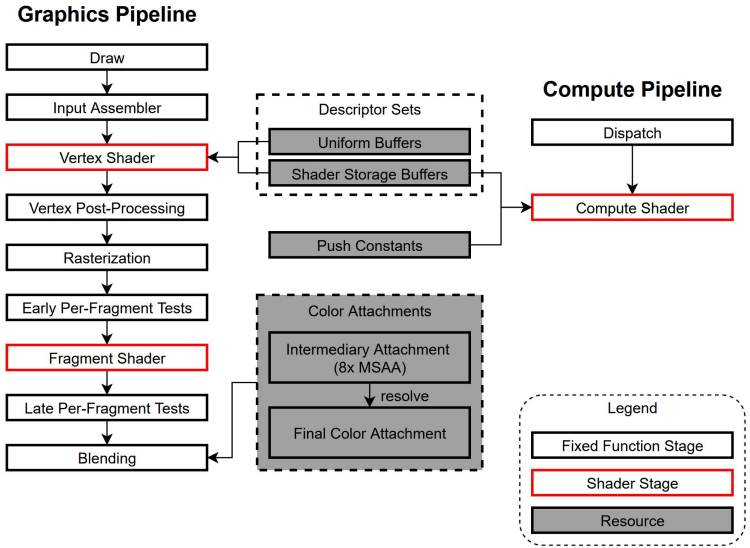
Pipeline

Etape de procesare:

1. **Compute Pipeline**
   - Compute Shader
   - Calcularea și actualizarea datelor fiecărei particule cu ajutorul unui Shader Storage Buffer
   - Folosește informațiile din Push Constants

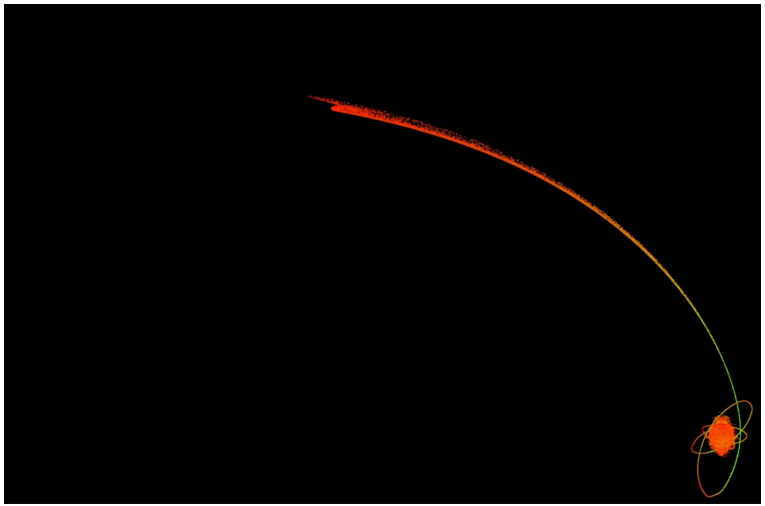
2. Graphics Pipeline
   - Vertex Shader
     → Folosește datele din Uniform Buffer și Shader Storage Buffer pentru a calcula poziția și culoarea

   - Fragment Shader
     → Color Attachment



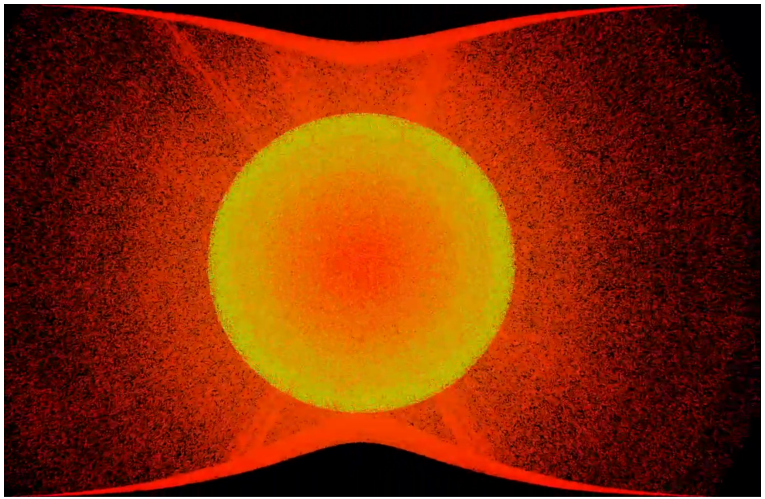
Performanță

Benchmark 1
simulează 8.388.608 de particule
concentrate în poziția centrului de atracție



Benchmark 1

Benchmark 2
simulează 8.388.608 de particule
cât mai răspândite, acoperind aproape
întreaga suprafață de vizualizare



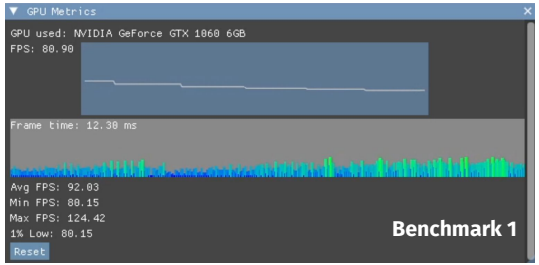
Benchmark 2

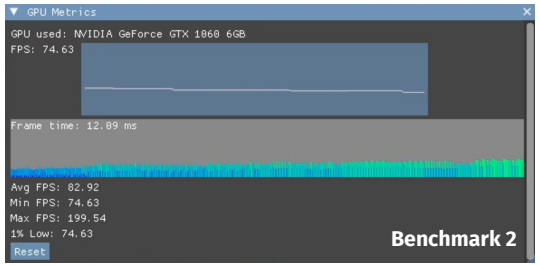
Performanță - GTX 1060 6GB

Configurație hardware: Intel i5-8400 4.0 GHz + 16 GB RAM DDR4 2400 MHz + NVIDIA GTX 1060 6 GB

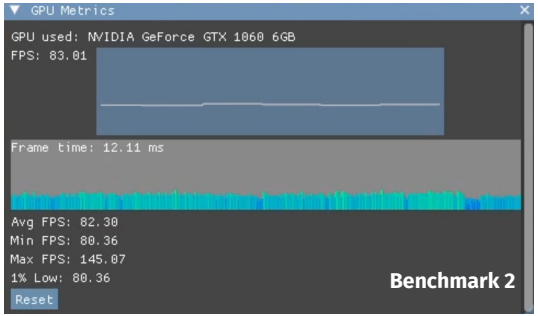
- Performanță mai bună pe Windows
- Experiența pe Ubuntu este mult mai stabilă
- Windows are spike-uri de performanță

Windows 11





Ubuntu 24.04.2

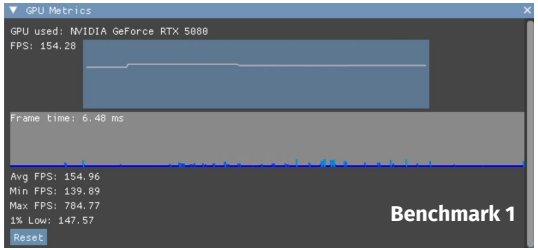


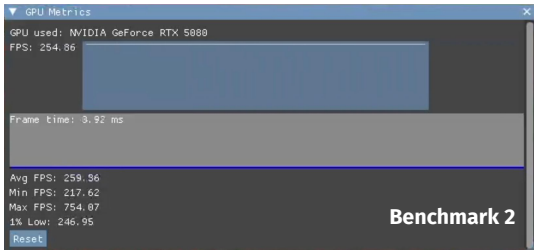
Performanță - RTX 5080 16 GB

Configurație hardware: AMD Ryzen 7 9800X3D 5.2 GHz + 32 GB RAM DDR5 6000 MHz + NVIDIA RTX 5080 16 GB GDDR7

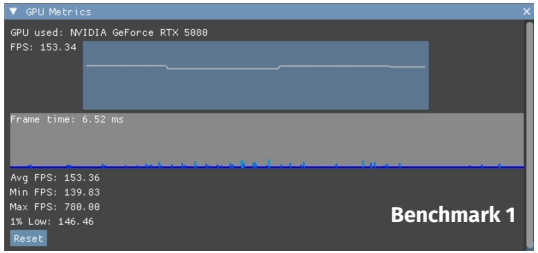
- Performanțe de 3 ori mai mari față de testele anterioare
- Noua configurație oferă un suport mult mai stabil
- Windows este capabil de a atinge performanțe mai bune, dar nu într-un mod constant

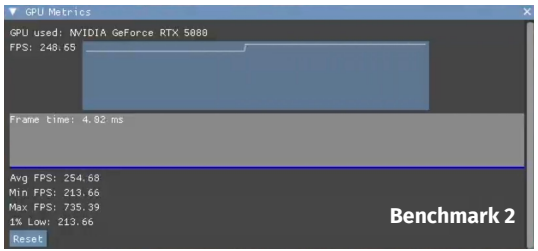
Windows 11





Ubuntu 24.04.2



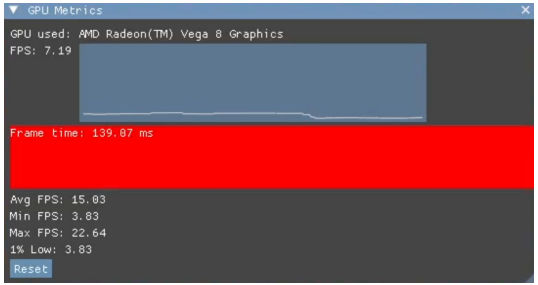


Performanță - placă video integrată

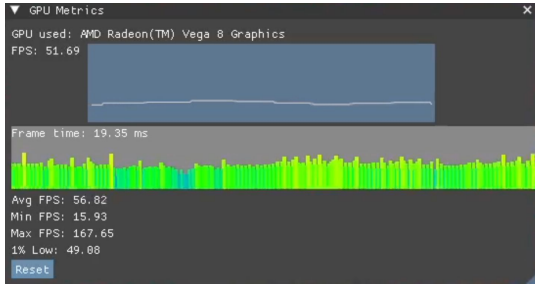
Configurație hardware: AMD Ryzen 5 3500U 3.7 GHz + 8GB RAM DDR4 2400 MHz + AMD Radeon Vega 8

- Nu poate procesa un flux mare de lucru și informații
- Poate funcționa bine doar cu o cantitate mică de date

8.388.608 particule



262.144 particule



Concluzii

- Nevoia unei plăci video dedicate sau integrate, cu suport complet pentru API-ul Vulkan
- Performanța aplicației depinde în mod direct de capacitatea dispozitivului hardware folosit
- Implementarea se va folosi de toate resursele disponibile pe care le poate oferi placa video
- Aplicația se adaptează odată cu hardware-ul, reflectând în mod direct performanța dispozitivului
- O placă video dedicată este esențială pentru rularea unui sistem complex cu volume mari de date

Configurație Hardware	Indicator	Benchmark 1		Benchmark 2	
		Windows 11	Ubuntu 24	Windows 11	Ubuntu 24
GTX 1060 6GB	AVG FPS	**92.03**	90.74	82.92	82.30
	MIN FPS	80.15	**88.72**	74.63	80.36
	MAX FPS	124.42	124.11	**199.54**	145.07
	1% Low FPS	80.15	**88.72**	74.63	80.36
RTX 5080 16GB	AVG FPS	154.96	153.36	**259.36**	254.68
	MIN FPS	139.89	139.83	**217.62**	213.66
	MAX FPS	**784.77**	780.00	754.07	735.39
	1% Low FPS	147.57	146.46	**246.95**	213.66

Configurație Hardware	Indicator	Particule	
		8.388.608	262.144
AMD Radeon Vega 8	AVG FPS	15.03	56.82
	MIN FPS	3.83	15.93
	MAX FPS	22.64	167.65
	1% Low FPS	3.83	49.08