

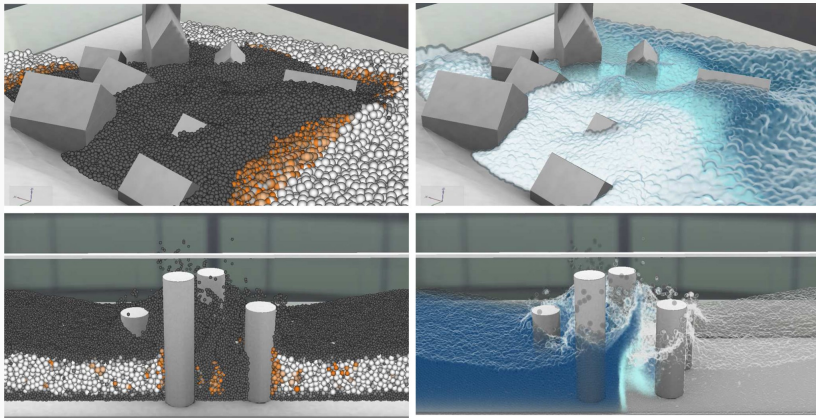
Techniki wizualizacji cieczy w czasie rzeczywistym z wykorzystaniem Rust oraz Vulkan API

Nikita Lysiuk

16 listopada 2025

Dlaczego symulacja cieczy?

- **Symulacja cieczy** to matematyczny opis zachowania płynów.
- Animacja ręczna jest trudna — zbyt dużo detali i zbyt duża dynamika.
- **Wizualizacja** jest większym wyzwaniem niż fizyka CPU.
- Rust + Vulkan = **kontrola**, wydajność i nowoczesny pipeline.



Rysunek 1: Przykład symulacji cieczy (SPH/SDF).

Krótko o fizyce.

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla) \mathbf{u} - \frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}$$

- **Adwekcja** $-(\mathbf{u} \cdot \nabla) \mathbf{u}$: istniejący ruch “przenosi” płyn dalej.
- **Ciśnienie** $-\frac{1}{\rho} \nabla p$: kontroluje objętość, rozpycha i ścisza płyn.
- **Lepkość** $\nu \nabla^2 \mathbf{u}$: wygładza ruch i redukuje energię.
- **Siły zewnętrzne** \mathbf{f} : grawitacja, turbulencje, wiatr.

Particle-based (SPH / PBF)

- Płyn jako chmura cząsteczek.
- Każda cząstka ma własną prędkość, gęstość i reakcję na siły.
- Naturalne zachowanie, łatwe kolizje, świetne do efektów VFX.

Grid-based (Euler)

- Przestrzeń podzielona na komórki (grid).
- W każdej komórce: prędkość, ciśnienie i inne parametry.
- Stabilne, dobre dla dużych objętości i realistycznych przepływów.

Podejście hybrydowe

- Particle do wyglądu, grid do stabilności i ciśnienia.
- Standard w solverach FLIP / APIC.

Co to jest Vulkan

- Niskopoziomowy interfejs do GPU — daje bezpośrednią kontrolę nad pamięcią i kolejkami.
- Nowoczesny, wydajniejszy następca OpenGL; zaprojektowany dla współczesnych kart.
- Cross-platform, w przeciwieństwie do DirectX.

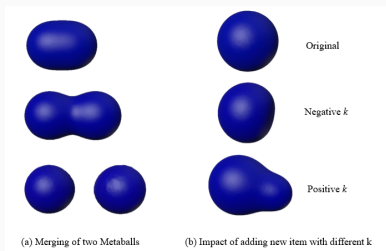
Dlaczego wybrałem Vulkan

- Chcę pełnej kontroli nad pipeline i zarządzaniem zasobami.
- Vulkan wymusza lepszą architekturę i bardziej świadome decyzje.
- Lepsza wydajność i większa przewidywalność niż OpenGL.

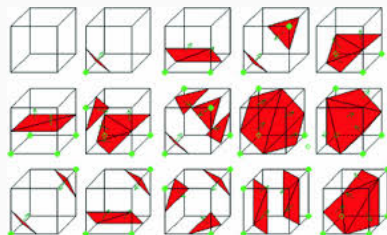
Techniki wizualizacji cieczy (które badam)

- **Metaballs / SDF** — reprezentacja płynu jako pola odległości,
- **Marching Cubes** — generowanie siatki powierzchni z pola SDF,
- **Screen-space fluid rendering** — efekt cieczy tworzony na etapie postprocessu,
- **Raymarching SDF** — wizualizacja bezpośrednio z pola odległości.

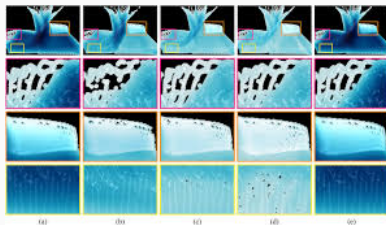
Przykładowe techniki wizualizacji cieczy



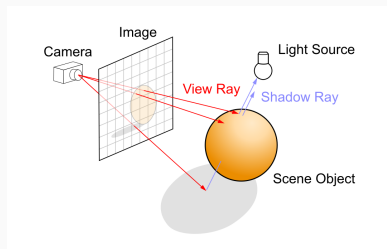
Metaballs / SDF



Marching Cubes



Screen-space rendering



Raymarching SDF

- Stworzyć prototypowy silnik 3D do wizualizacji cieczy.
- Opracować CPU-solver w Rust z wykorzystaniem Rayon (SPH/PBF).
- Zbudować pipeline graficzny w Vulkan do generowania powierzchni cieczy.
- Uzyskać interaktywny rendering w czasie rzeczywistym.

- Backend fizyki oparty o compute shaders.
- CUDA i porównanie wydajności z CPU.
- Techniki Machine Learning do przewidywania zachowania cieczy.
- Większe sceny, interakcje, kolizje i rendering objętościowy.