Uniwersytet Wrocławski

Programowanie Równoległe
Wykład, 27.01.2014

CUDA praktycznie 4
Aplikacje (system cząsteczek)

Maciej Matyka
Instytut Fizyki Teoretycznej

Plan CUDA w praktyce

- Wykład 1: CUDA w praktyce
- Wykład 2: Cuda + Opengl

Wykład 3: Cuda + Opengl Interoperability

Wykład 4: Aplikacje (systemy cząsteczek)

Przegląd

Co potrafimy:

- trzymać dane (punkty) na GPU

Przegląd

Co potrafimy:

- trzymać dane (punkty) na GPU
- rysować wprost z GPU (CUDA – OpenGL Interoperability)

Przegląd

Co potrafimy:

- trzymać dane (punkty) na GPU
- rysować wprost z GPU (CUDA – OpenGL Interoperability)
- operować na nich (kernele)

Przegląd

Inne tryby rysowania (**point sprites**)

http://youtu.be/zSKAQ3AyKtU
http://youtu.be/gDPnWgQaMR8

Co potrafimy:

- trzymać dane (punkty) na GPU
- rysować wprost z GPU (CUDA – OpenGL Interoperability)
- operować na nich (kernele)

Przegląd

Inne tryby rysowania (**point sprites**)

http://youtu.be/zSKAQ3AyKtU
http://youtu.be/gDPnWgQaMR8

Co potrafimy:

- trzymać dane (punkty) na GPU
- rysować wprost z GPU (CUDA – OpenGL Interoperability)
- operować na nich (kernele)

nbody

http://youtu.be/MkcTNZBJYKI
http://youtu.be/ztwkXq4Hj7Q
http://youtu.be/byI9yhITDsM
http://youtu.be/PrIk6dKcdoU
http://youtu.be/Cd9cBlvfjow

Przegląd

Inne tryby rysowania (**point sprites**)

http://youtu.be/zSKAQ3AyKtU
http://youtu.be/gDPnWgQaMR8

Co potrafimy:

- trzymać dane (punkty) na GPU
- rysować wprost z GPU (CUDA – OpenGL Interoperability)
- operować na nich (kernele)

nbody

http://youtu.be/MkcTNZBJYKI
http://youtu.be/ztwkXq4Hj7Q
http://youtu.be/byI9yhITDsM
http://youtu.be/PrIk6dKcdoU
http://youtu.be/Cd9cBlvfjow

Flocking & particles

http://youtu.be/7cjorOe810o

Przegląd

Inne tryby rysowania (**point sprites**)

http://youtu.be/zSKAQ3AyKtU
http://youtu.be/gDPnWgQaMR8

Co potrafimy:

- trzymać dane (punkty) na GPU
- rysować wprost z GPU (CUDA – OpenGL Interoperability)
- operować na nich (kernele)

nbody
http://youtu.be/MkcTNZBJYKI
http://youtu.be/ztwkXq4Hj7Q
http://youtu.be/byI9yhITDsM
http://youtu.be/PrIk6dKcdoU
http://youtu.be/Cd9cBlvfjow

Flocking & particles
http://youtu.be/7cjorOe810o

Spring-mass system
http://youtu.be/_FTlkslaZ4A
http://youtu.be/LiBrGZE77DI

Przegląd

Inne tryby rysowania (**point sprites**)

http://youtu.be/zSKAQ3AyKtU
http://youtu.be/gDPnWgQaMR8

Co potrafimy:

- trzymać dane (punkty) na GPU
- rysować wprost z GPU (CUDA – OpenGL Interoperability)
- operować na nich (kernele)

nbody
http://youtu.be/MkcTNZBJYKI
http://youtu.be/ztwkXq4Hj7Q
http://youtu.be/byI9yhITDsM
http://youtu.be/PrIk6dKcdoU
http://youtu.be/Cd9cBlvfjow

Granular matter
http://youtu.be/9yLU3aNTdwc
http://youtu.be/-QJ4bAtS2rk

Flocking & particles
http://youtu.be/7cjorOe810o

Spring-mass system
http://youtu.be/_FTlkslaZ4A
http://youtu.be/LiBrGZE77DI

Przegląd

Inne tryby rysowania (**point sprites**)

http://youtu.be/zSKAQ3AyKtU
http://youtu.be/gDPnWgQaMR8

Co potrafimy:

- trzymać dane (punkty) na GPU
- rysować wprost z GPU (CUDA – OpenGL Interoperability)
- operować na nich (kernele)

Fluids

http://youtu.be/TU0fa8FUugg

nbody

http://youtu.be/MkcTNZBJYKI
http://youtu.be/ztwkXq4Hj7Q
http://youtu.be/byI9yhITDsM
http://youtu.be/PrIk6dKcdoU
http://youtu.be/Cd9cBlvfjow

Granular matter

http://youtu.be/9yLU3aNTdwc
http://youtu.be/-QJ4bAtS2rk

Flocking & particles

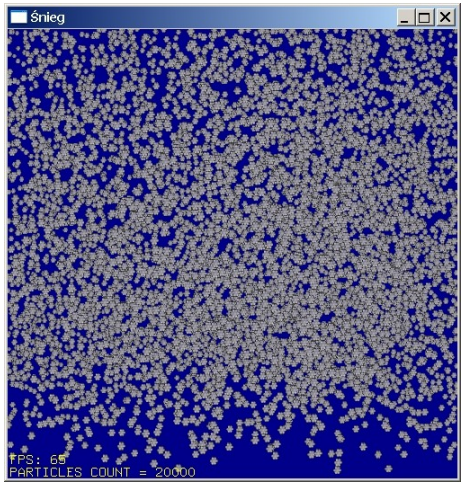
http://youtu.be/7cjorOe810o

Spring-mass system

http://youtu.be/_FTlkslaZ4A
http://youtu.be/LiBrGZE77DI

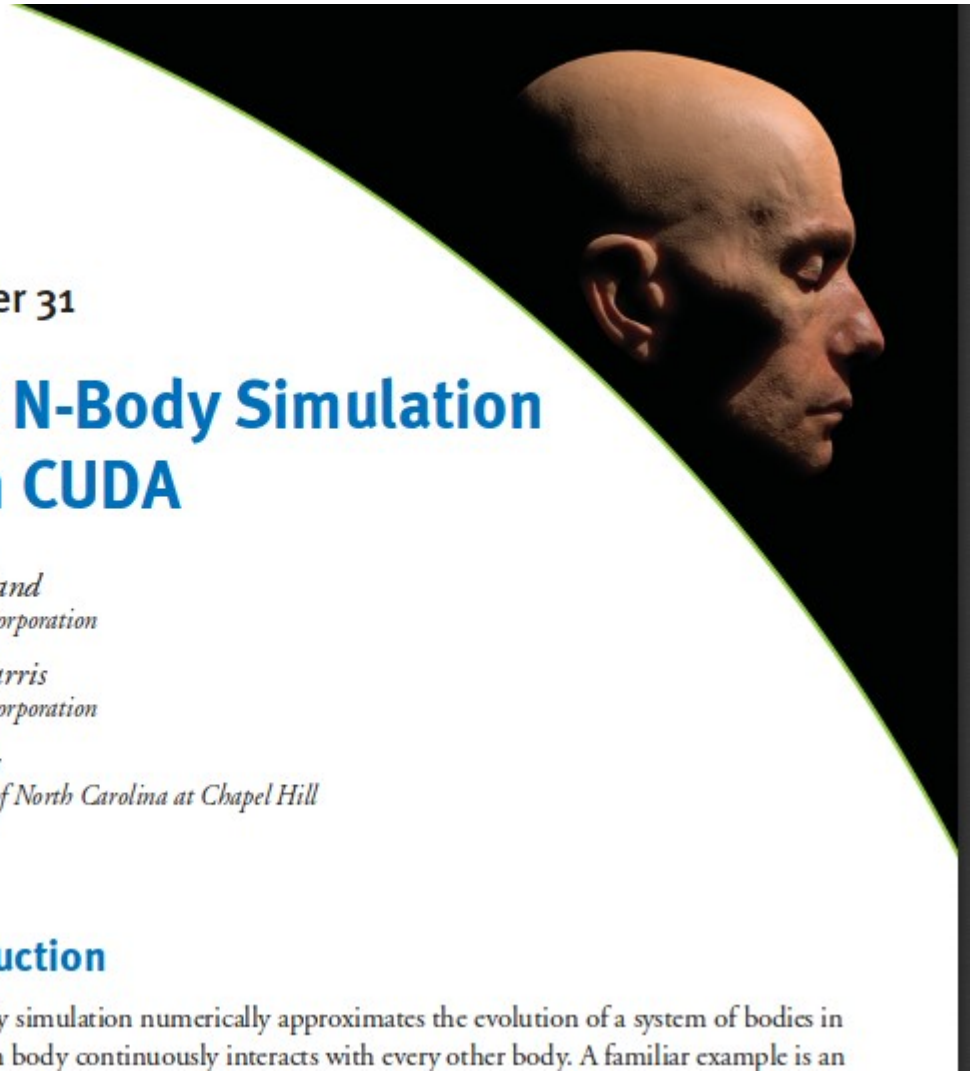
Point sprites

http://cpp0x.pl/kursy/Kurs-OpenGL-C++/Systemy-czastek/238




```
// włączenie sprajtów punktowych
glEnable( GL_POINT_SPRITE );

// włączenie tablic wierzchołków
glEnableClientState( GL_VERTEX_ARRAY );

// zdefiniowanie danych tablic wierzchołków
glVertexPointer( 3, GL_FLOAT, sizeof( SnowFlake ), SnowFlakes );

// narysowanie danych zawartych w tablicach
glDrawArrays( GL_POINTS, 0, flake_count );
```

n-body



Chapter 31

Fast N-Body Simulation with CUDA

Lars Nyland
NVIDIA Corporation

Mark Harris
NVIDIA Corporation

Jan Prins
University of North Carolina at Chapel Hill

31.1 Introduction

An N-body simulation numerically approximates the evolution of a system of bodies in which each body continuously interacts with every other body. A familiar example is an

GPU Gems 3, Chapter 31. Fast N-Body Simulation with CUDA

n-body

Listing 31-1. Updating Acceleration of One Body as a Result of Its Interaction with Another Body

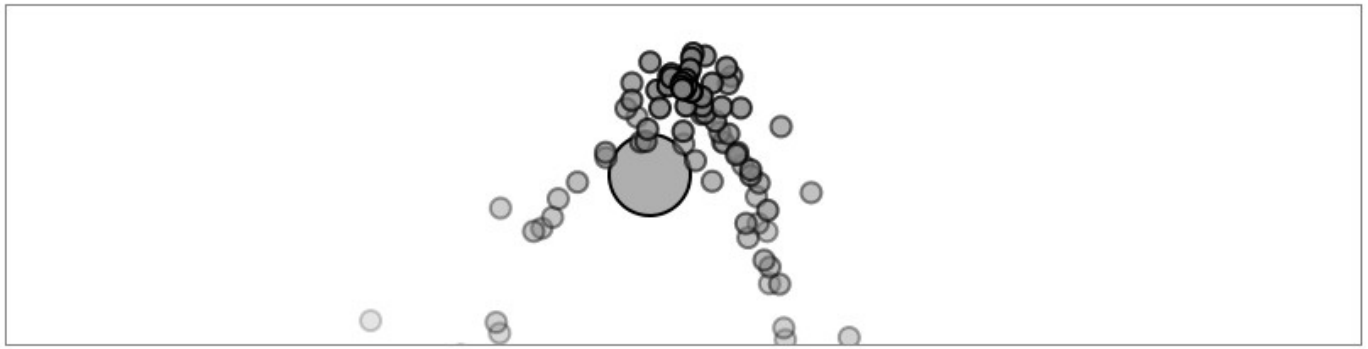
```
__device__ float3
bodyBodyInteraction(float4 bi, float4 bj, float3 ai)
{
  float3 r;

  // r_ij  [3 FLOPS]
  r.x = bj.x - bi.x;
  r.y = bj.y - bi.y;
  r.z = bj.z - bi.z;

  // distSqr = dot(r_ij, r_ij) + EPS^2  [6 FLOPS]
  float distSqr = r.x * r.x + r.y * r.y + r.z * r.z + EPS2;

  // invDistCube =1/distSqr^(3/2)  [4 FLOPS (2 mul, 1 sqrt, 1 inv)]
  float distSixth = distSqr * distSqr * distSqr;
  float invDistCube = 1.0f/sqrtf(distSixth);

  // s = m_j * invDistCube [1 FLOP]
  float s = bj.w * invDistCube;

  // a_i =  a_i + s * r_ij [6 FLOPS]
  ai.x += r.x * s;
  ai.y += r.y * s;
  ai.z += r.z * s;

  return ai;
}
```

Particle system

http://natureofcode.com/book/chapter-4-particle-systems/

THE **NATURE** OF CODE by Daniel Shiffman Buy this book

`Particle` class, which hasn't changed.



RESET PAUSE

Example 4.7: ParticleSystem with repeller

`ParticleSystem ps;`	One ParticleSystem
`Repeller repeller;`	One repeller

```
void setup() {
  size(640,360);
  ps = new ParticleSystem(new PVector(width/2,50));
  repeller = new Repeller(width/2-20,height/2);
}
```

Granular matter

$$\xi = \max(\,0,\, R_1 + R_2 - \|\vec{x}_2 - \vec{x}_1\|)\,, \tag{1}$$

$$\overrightarrow{N} = \frac{x_2 - x_1}{\|x_2 - x_1\|}\,. \tag{2}$$

$$\overrightarrow{V} = \vec{v}_1 - \vec{v}_2\,, \tag{3}$$

$$\dot{\xi} = \overrightarrow{V} \cdot \overrightarrow{N}\,, \tag{4}$$

$$\overrightarrow{V}_t = \overrightarrow{V} - \dot{\xi}\overrightarrow{N}\,. \tag{5}$$

$$\overrightarrow{F}_n = f_n \overrightarrow{N}\,, \tag{6}$$

$$f_n + k_d \xi^{\alpha} \dot{\xi} + k_r \xi^{\beta} = 0\,, \tag{7}$$

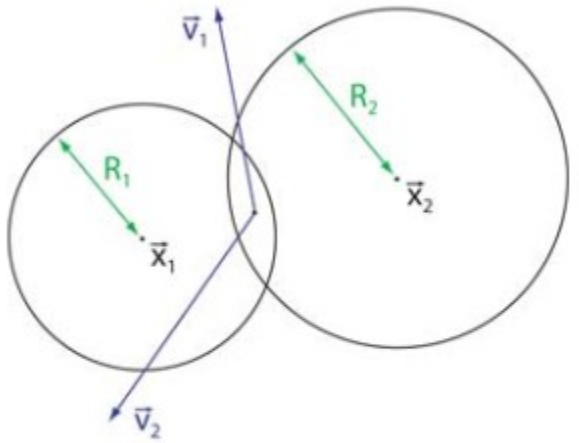


Figure 2: *Quantities used in collision modeling.*

Granular matter

$$\xi = \max(\,0,\, R_1 + R_2 - \|\vec{x}_2 - \vec{x}_1\|\,), \qquad (1)$$

$$\vec{N} = \frac{x_2 - x_1}{\|x_2 - x_1\|}. \qquad (2)$$

$$\vec{V} = \vec{v}_1 - \vec{v}_2, \qquad (3)$$

$$\dot{\xi} = \vec{V} \cdot \vec{N}, \qquad (4)$$

$$\vec{V}_t = \vec{V} - \dot{\xi}\vec{N}. \qquad (5)$$

$$\vec{F}_n = f_n \vec{N}, \qquad (6)$$

$$f_n + k_d \xi^{\alpha} \dot{\xi} + k_r \xi^{\beta} = 0, \qquad (7)$$

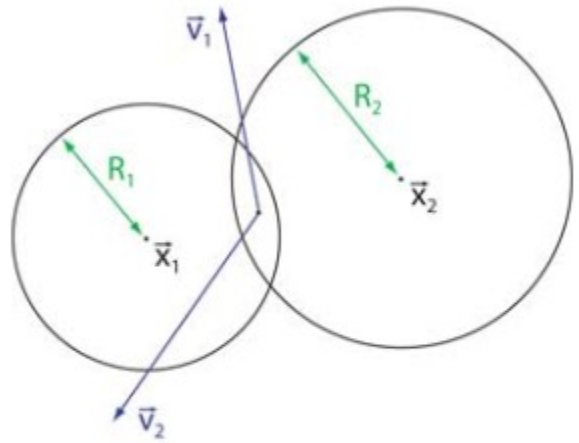


Figure 2: *Quantities used in collision modeling.*

N. Bell, Y. Yu and P. J. Mucha / Particle-Based Simulation of Granular Materials

Figure 7: *Structures swept away by an avalanche.*

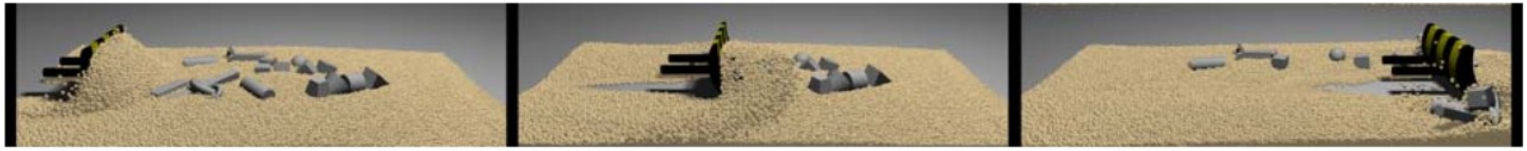


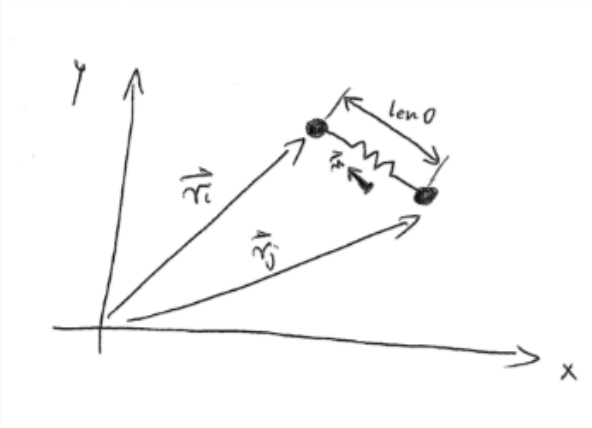
Figure 8: *A bulldozer clears the remains of the avalanche.*

Spring-mass system

Punkt zawieszony na linie

Siła między punktami: sprężystość + tłumienie

$$\mathbf{f}(i,j) = -ks^{*}(|\mathbf{ri}-\mathbf{rj}| - len0)^{*}\mathbf{n} + kd^{*}|\mathbf{v2}-\mathbf{v1}|^{*}\mathbf{n};$$



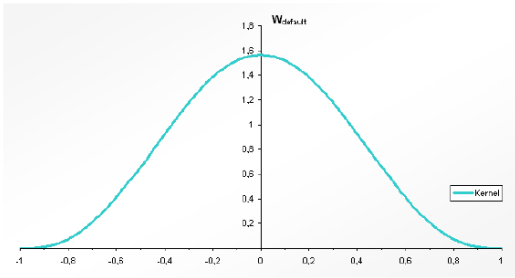
http://www.pagines.ma1.upc.edu/~susin/files/AdvancedCharacterPhysics.pdf

- Podejście Lagrange’a (ruchome punkty)
- Każda cząstka reprezentuje pewną objętość cieczy
- Wielkości fizyczne interpolowane po sąsiadach

$$A_i = \sum_{j=1}^{N} m_j \frac{A_j}{\rho_j} W_{ij}$$

- Gdzie Wij jest tytułowym „rozmyciem” np.

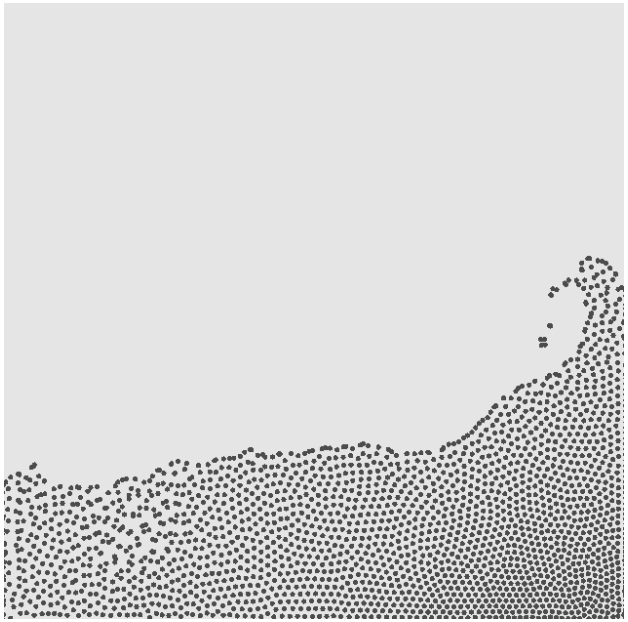
$$W_{default}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} \left(h^2 - \|\mathbf{r}\|^2\right)^3 & 0 \le \|\mathbf{r}\| \le h \\ 0 & \|\mathbf{r}\| > h, \end{cases}$$



- 1. Lagrangian Fluid Dynamics, Using Smoothed Particle Hydrodynamics, Micky Kelager, 2006
- 2. http://pl.wikipedia.org/wiki/SPH)

- Inicjalizacja pozycji i prędkości cząstek SPH

- Obliczenie gęstości cząstkek (z kerneli rozmycia)
- Obliczenie sił działających na cząstkę
  - ciśnienie
  - lepkość,
  - siły powierzchniowe
- Przesunięcie cząstek
- (najprościej, ale nie najlepiej
- schemat Eulera...)



- $v = \dot{v} + (\mathbf{F/m})*dt$
- $p = p + v * dt$

http://fluids3.com/

http://youtu.be/hUezoHa1ZF4

- (SPH w działaniu)

Fluids - aerodynamika

Proste przepływy można przybliżyć równaniem Laplace'a:

$$\nabla.\mathbf{v} = \nabla.\nabla\phi = \nabla^2\phi = 0. \qquad (1)$$

$$\phi = \frac{a}{2\pi}\ln r; \quad \mathbf{v}_r = \frac{a}{2\pi r}; \quad \mathbf{v}_\theta = 0; \quad \mathbf{v}_z = 0. \qquad (2)$$

For a sink the constant a is set negative. A vortex at the origin with strength b is given by:

$$\phi = \frac{b}{2\pi}\theta; \quad \mathbf{v}_r = 0; \quad \mathbf{v}_\theta = \frac{b}{2\pi r}; \quad \mathbf{v}_z = 0. \qquad (3)$$

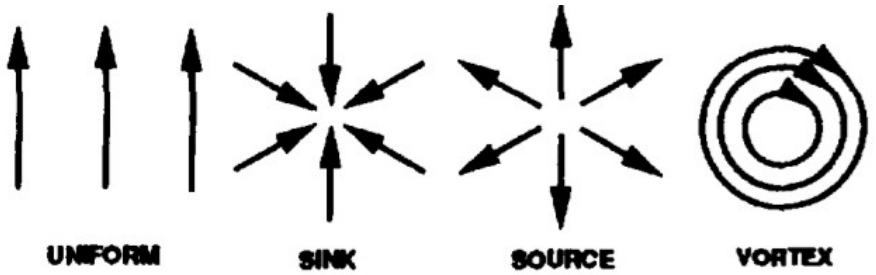


Figure 1. A schematic diagram of the flow primitives.

Wejchert, J. and Haumann, Animation Aerodynamics, Computer Graphics (ACM)

Fluids - aerodynamika

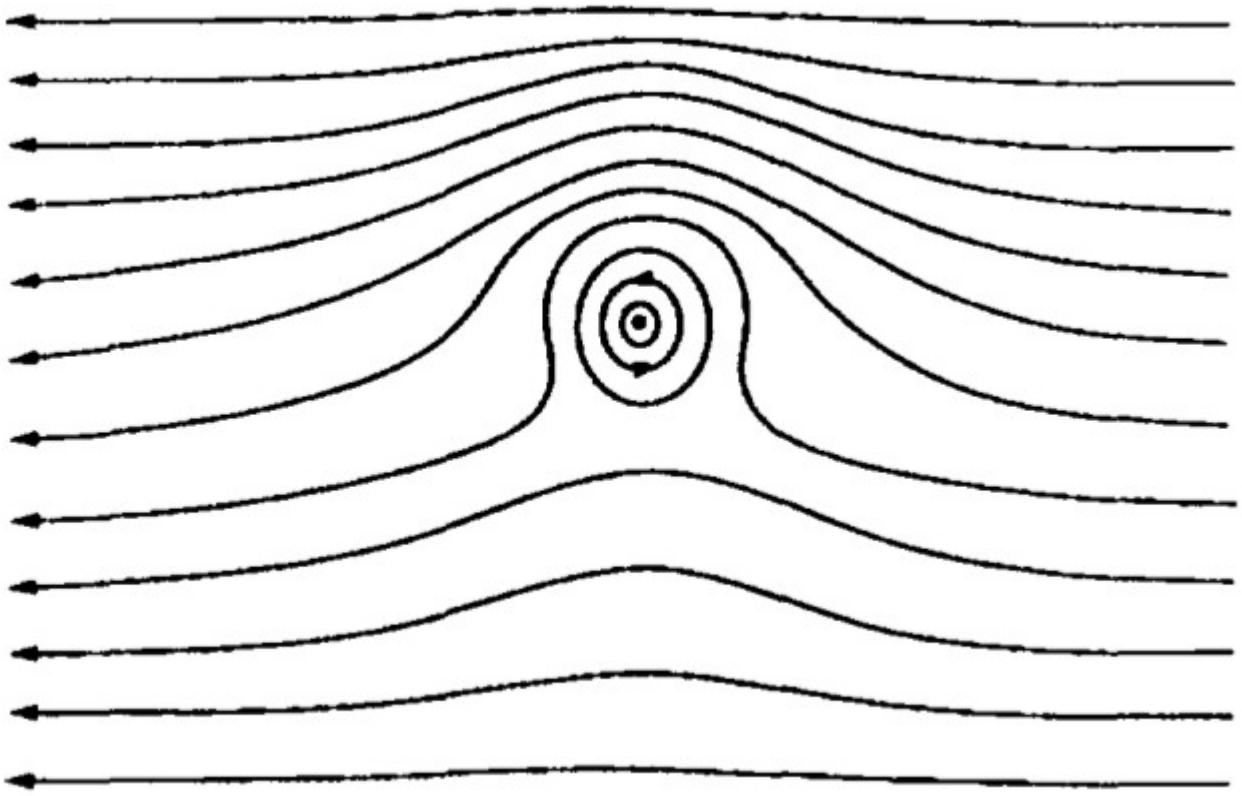
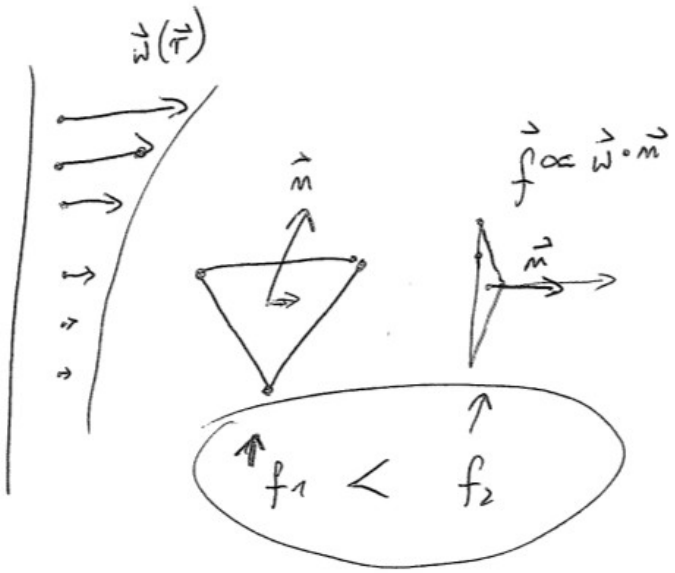


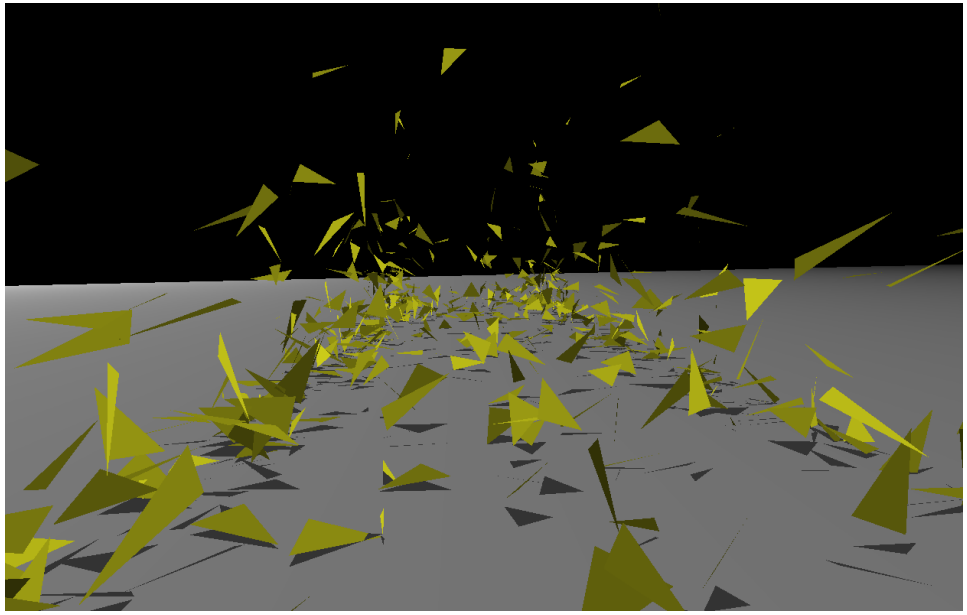
Figure 2. The addition of uniform and vortex flow.

Wejchert, J. and Haumann, Animation Aerodynamics, Computer Graphics (ACM)

Fluids - aerodynamika

Siła proporcjonalna do rzutu wektora przepływu na wektor normalny ścianki:





Wejchert, J. and Haumann, Animation Aerodynamics, Computer Graphics (ACM)

Zadanie na dzisiaj

Wybierz jeden z przedstawionych modeli.
Zaimplementuj proof-of-concept w CUDA.

Zadanie na 2 tygodnie.