Uniwersytet Wrocławski

Modelowanie komputerowe dynamiki płynów

2011-12-05

Maciej Matyka
Instytut Fizyki Teoretycznej

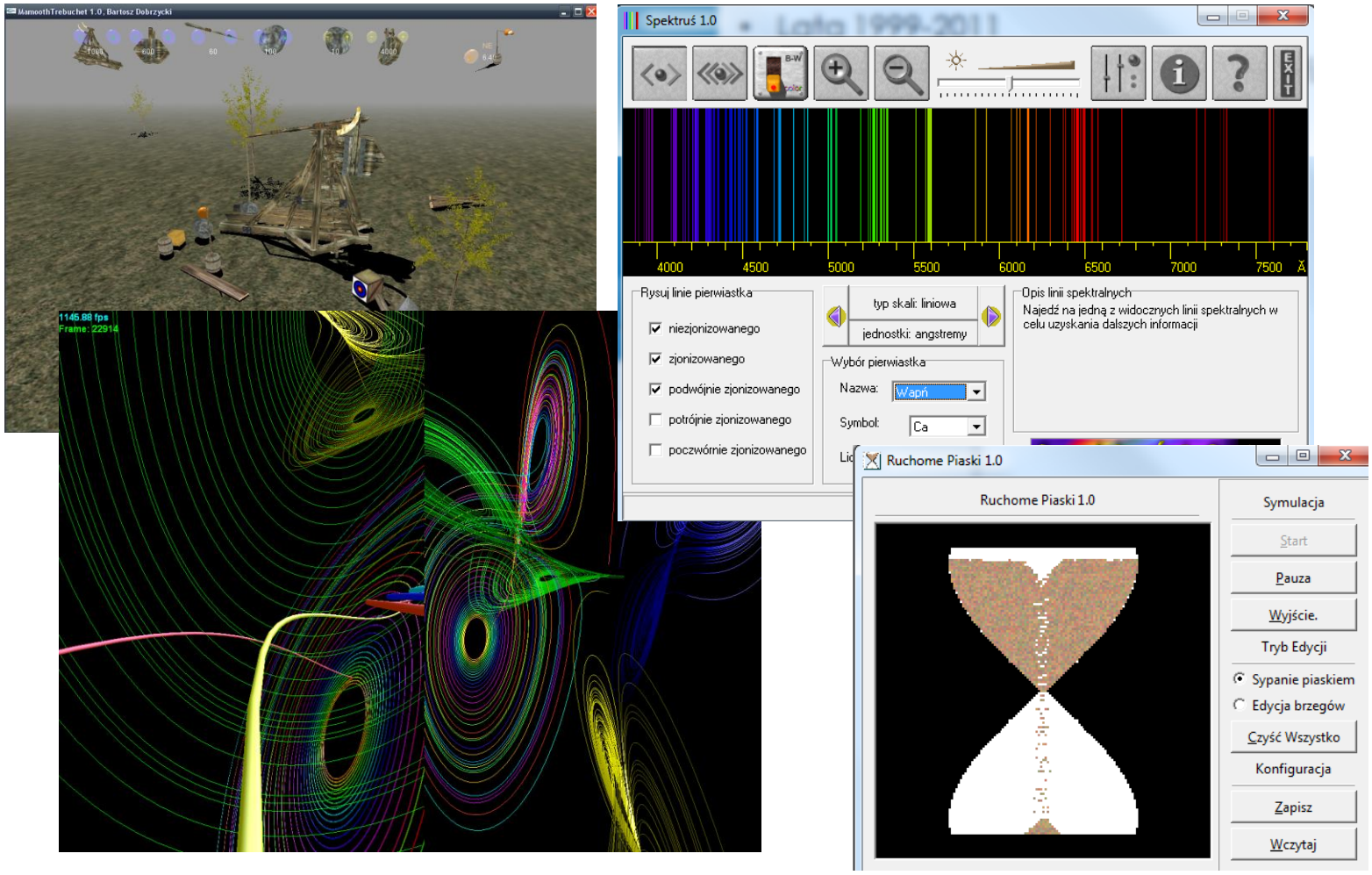
Uniwersytet Wrocławski

Hydrodynamika

http://www.realflow.com/

Konkursy na wydziale fizyki

- Lata 1999-2011
- Oprogramowanie popularyzujące fizykę



Uniwersytet Wrocławski

Konkurs

na oprogramowanie popularyzujące fizykę

011001
fizbit

Instutut Fizyki Teoretycznej ma zaszczyt zaprosić studentów **I-V** roku **Wydziału Fizyki i Astronomii UWr** do konkursu na oprogramowanie popularyzujące fizykę

1. Dowolna technologia i forma (program, animacja, strona www, materiał youtube/vimeo etc.)

2. Prace należy dostarczyć do **30 Kwietnia 2012** do **Macieja Matyki** na adres: maq@ift.uni.wroc.pl lub maciej.matyka@gmail.com

3. Ogłoszenie zwycięzców nastąpi do **31 Maja 2012**. Zwycięzcy będą mieli szansę zaprezentować swoje prace w trakcie rozdania nagród.

4. Przewidujemy **nagrody finansowe**.

5. Programy z konkursów z poprzednich lat: http://www.ift.uni.wroc.pl/fkomp/?kat=prog

6. Komisja konkursowa składa się z osób związanych z fizyką komputerową w Instytucie Fizyki Teoretycznej.

Zapraszamy do udziału!

Uniwersytet Wrocławski

Jak wyświetlić dane symulacji na ekranie?

- Animacja w czasie rzeczywistym (buforowanie)
- Możliwość interakcji z użytkownikiem
- Prostota rozwiązania
- **OpenGL**
- Przenośność kodu
  - Windows
  - Linux
  - urządzenia mobilne
- Integracja (QT, WinAPI, SDL etc.)
- **GL Utility Toolkit (GLUT)**
  - Zestaw procedur upraszczających życie

Uniwersytet Wrocławski

Graficzne hello world

- Zestaw punktów materialnych na ekranie (**OpenGL**)
- Przyspieszenie grawitacyjne
- (punkty odbijają się od podłoża)

- Dla każdego punktu:
  - vy = vy + g*dt // prędkość
  - py = py + vy * dt // pozycja
  - Jeśli y<0 to // kolizja

    y=-y;

    vy = vy;

Otwarcie okna OpenGL

- Minimalny kod w GLUT (GL Utility Toolkit)
- Schemat:

```
int main(int argc, char **argv)
{
  glutInit(&argc, argv);

  glutInitDisplayMode(GLUT_DEPTH |
  GLUT_DOUBLE | GLUT_RGBA);

  glutInitWindowPosition(100,100);
  glutInitWindowSize(320,320);
  glutCreateWindow("CUDA Hello World
  GFX");

    C1

  // register callbacks
  glutDisplayFunc(renderScene);
  glutReshapeFunc(changeSize);
  glutIdleFunc(idleFunction);

  // enter GLUT event processing cycle
  glutMainLoop();
}
```

```
void changeSize(int w, int h)
{
  float ratio = 1.0 * w / h;
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glViewport(0, 0, w, h);
  gluOrtho2D(-1,1,-1,1);
  glMatrixMode(GL_MODELVIEW);
}
```

```
void idleFunction(void)
{
    C2
    glutPostRedisplay();
}
```

```
void renderScene(void)
{
  C3
    glutSwapBuffers();
}
```

Więcej: http://www.lighthouse3d.com/opengl/

Otwarcie okna OpenGL

- Minimalny kod w GLUT (GL Utility Toolkit)
- Schemat:

```
int main(int argc, char **argv)
{
  glutInit(&argc, argv);

  glutInitDisplayMode(GLUT_DEPTH |
  GLUT_DOUBLE | GLUT_RGBA);

  glutInitWindowPosition(100,100);
  glutInitWindowSize(320,320);
  glutCreateWindow("CUDA Hello World
  GFX");

    C1

  // register callbacks
  glutDisplayFunc(renderScene);
  glutReshapeFunc(changeSize);
  glutIdleFunc(idleFunction);

  // enter GLUT event processing cycle
  glutMainLoop();
}
```

```
void changeSize(int w, int h)
{
  float ratio = 1.0 * w / h;
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glViewport(0, 0, w, h);
  gluOrtho2D(-1,1,-1,1);
  glMatrixMode(GL_MODELVIEW);
}
```

```
void idleFunction(void)
{
    C2
  glutPostRedisplay();
}
```

```
void renderScene(void)
{
  C3
  glutSwapBuffers();
}
```

Więcej: http://www.lighthouse3d.com/opengl/

Otwarcie okna OpenGL

- Minimalny kod w GLUT (GL Utility Toolkit)
- Schemat:

```
int main(int argc, char **argv)
{
  glutInit(&argc, argv);

  glutInitDisplayMode(GLUT_DEPTH |
  GLUT_DOUBLE | GLUT_RGBA);

  glutInitWindowPosition(100,100);
  glutInitWindowSize(320,320);
  glutCreateWindow("CUDA Hello World
  GFX");

    C1

  // register callbacks
  glutDisplayFunc(renderScene);
  glutReshapeFunc(changeSize);
  glutIdleFunc(idleFunction);

  // enter GLUT event processing cycle
  glutMainLoop();
}
```

```
void changeSize(int w, int h)
{
  float ratio = 1.0 * w / h;
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glViewport(0, 0, w, h);
  gluOrtho2D(-1,1,-1,1);
  glMatrixMode(GL_MODELVIEW);
}
```

```
void idleFunction(void)
{
    C2

    glutPostRedisplay();
}
```

```
void renderScene(void)
{
  C3
      glutSwapBuffers();
}
```

Więcej: http://www.lighthouse3d.com/opengl/

Otwarcie okna OpenGL

- Minimalny kod w GLUT (GL Utility Toolkit)
- Schemat:

```
int main(int argc, char **argv)
{
  glutInit(&argc, argv);

  glutInitDisplayMode(GLUT_DEPTH |
  GLUT_DOUBLE | GLUT_RGBA);

  glutInitWindowPosition(100,100);
  glutInitWindowSize(320,320);
  glutCreateWindow("CUDA Hello World
  GFX");

    C1

  // register callbacks
  glutDisplayFunc(renderScene);
  glutReshapeFunc(changeSize);
  glutIdleFunc(idleFunction);

  // enter GLUT event processing cycle
  glutMainLoop();
}
```

```
void changeSize(int w, int h)
{
  float ratio = 1.0 * w / h;
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glViewport(0, 0, w, h);
  gluOrtho2D(-1,1,-1,1);
  glMatrixMode(GL_MODELVIEW);
}
```

```
void idleFunction(void)
{
    C2
    glutPostRedisplay();
}
```

```
void renderScene(void)
{
  C3
  glutSwapBuffers();
}
```

Więcej: http://www.lighthouse3d.com/opengl/

Uniwersytet Wrocławski

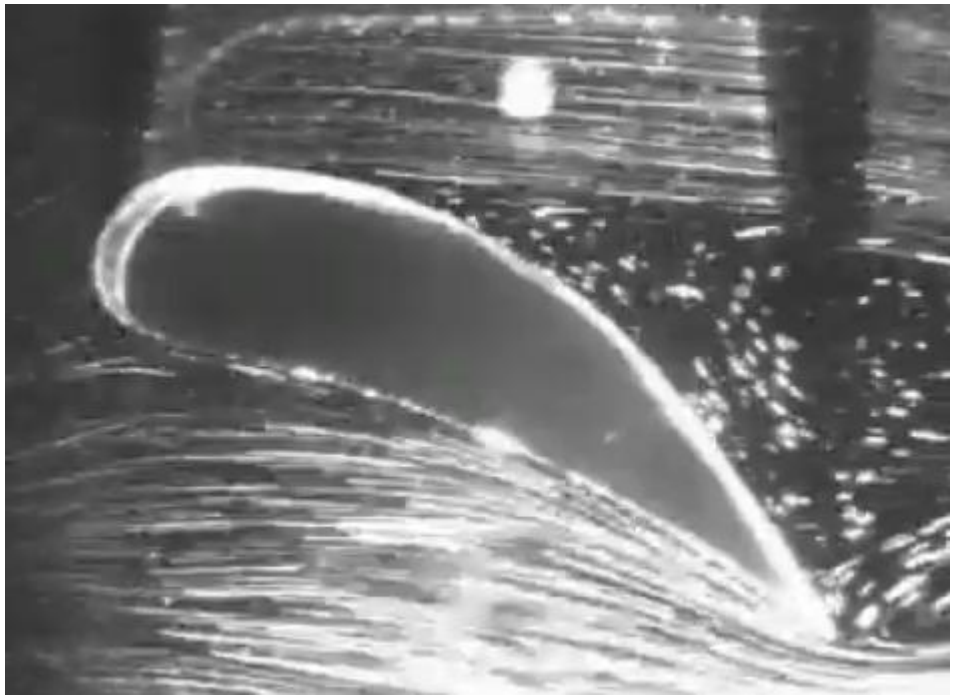
Konstrukcja

- C1 – ustawienie współrzędnych punktów
- C2 – przesunięcie punktów w nowe pozycje
- C3 – narysowanie punktu na ekranie

(przykład Visual C++)

Uniwersytet Wrocławski

Symulacje komputerowe cieczy



http://interactiveflows.com

Równania Naviera--Stokesa

- Równania Naviera-Stokesa dla cieczy nieściśliwej:

$$\overbrace{\rho\left(\underbrace{\frac{\partial \mathbf{u}}{\partial t}}_{\text{Unsteady acceleration}} + \underbrace{\mathbf{u}\cdot\nabla\mathbf{u}}_{\text{Convective acceleration}}\right)}^{\text{Inertia}} = \underbrace{-\nabla p}_{\text{Pressure gradient}} + \underbrace{\mu\nabla^2\mathbf{u}}_{\text{Viscosity}} + \underbrace{\mathbf{f}}_{\text{Other forces}}$$

$$\underbrace{\nabla\cdot\mathbf{u} = 0}_{\text{Continuity equation}}$$

- Nieliniowe równanie różniczkowe drugiego rzędu
- Równanie ciągłości przy założeniu stałej gęstości

http://en.wikipedia.org/wiki/Navier-Stokes equations

Uniwersytet Wrocławski

Metody rozwiązania

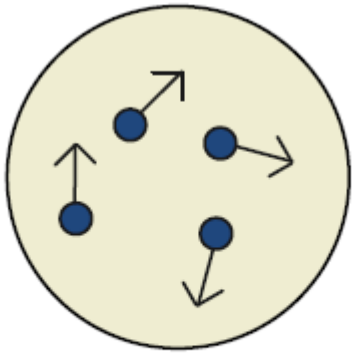
Skala makro (continuum)

- Navier–Stokes Equations (NSE)
  - Finite Difference Method (FDM)
  - Finite Volume Method (FVM)
  - Finite Element Method (FEM)
- **Smoothed Particle Hydrodynamics** (SPH)
- Dissipative Particle Dynamics (DPD)
- **The Lattice Boltzmann Method (LBM)**
- **Lattice Gas Automata** (LGA)
- **Molecular Dynamics** (MD)

Skala mikro (poziom atomowy)

Dynamika molekularna

- Duża ilość atomów



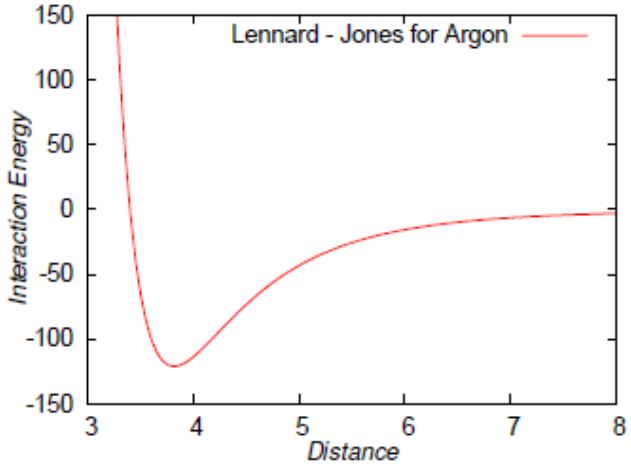
MD

- równania ruchu
- oddziaływanie atomu z atomem
- Skala mikroskopowa

- koncepcja Demona Laplace'a ?
- Tylko małe układy

- Potencjał L-J

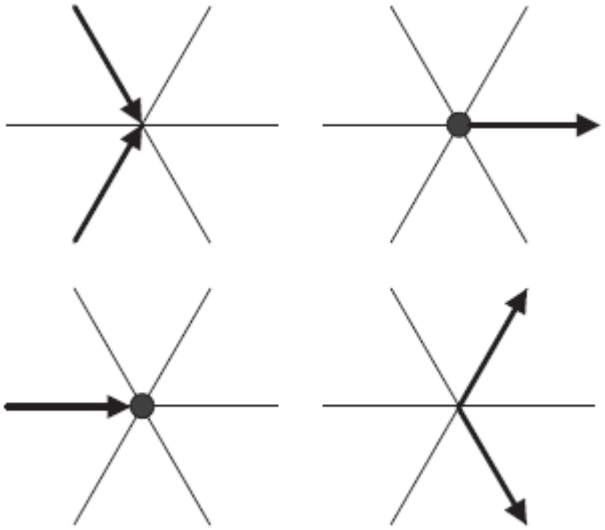
$$V(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$$



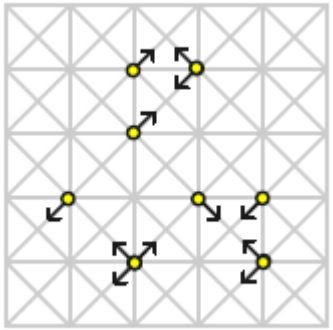
Allen, M.P et al., "Computer Simulation of Liquids", Oxford Univ. Press.

Model gazu sieciowego LGA

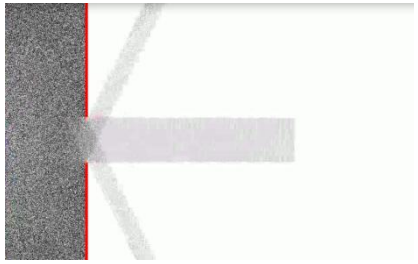
- Uproszczenie: model gazu sieciowego opisany przez U. Frisha, B. Hasslachera i Y. Pomeau gaz FHP



Rys. Kolizje w modelu FHP5



1) Transport
2) Kolizje



U. Frish, B. Hasslacher, Y. Pomeau 1986 *Lattice-Gas Automata for the Navier-Stokes Equation*, Phys. Rev. Lett. 56, 1505–1508.

Matyka, M. and Koza, Z., *Spreading of a density front in the Kuentz-Lavallee model of porous media* J. Phys. D: Appl. Phys. 40, 4078-4083 (2007)

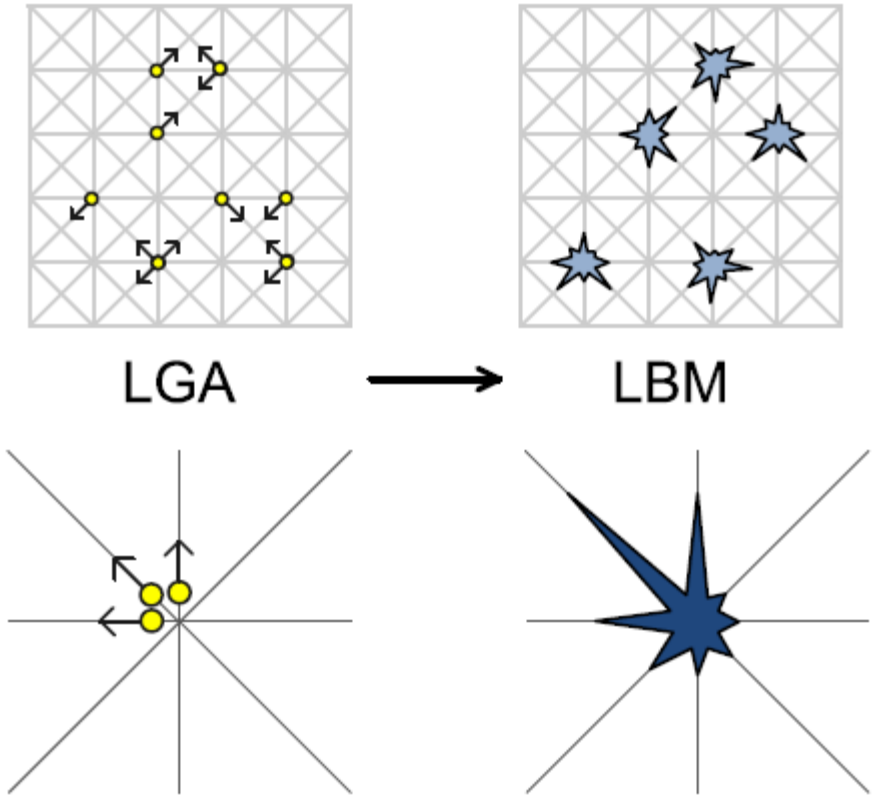
Uniwersytet Wrocławski

Model gazu sieciowego LGA

- Model LGA nie jest doskonały:
  - szum statystyczny
  - wymaga uśredniania w czasie
  - wymaga uśredniania w przestrzeni
  - wymaga dużych siatek (pamięć)
- Rzadko używany w „poważnych” aplikacjach

Metoda gazu sieciowego Boltzmanna

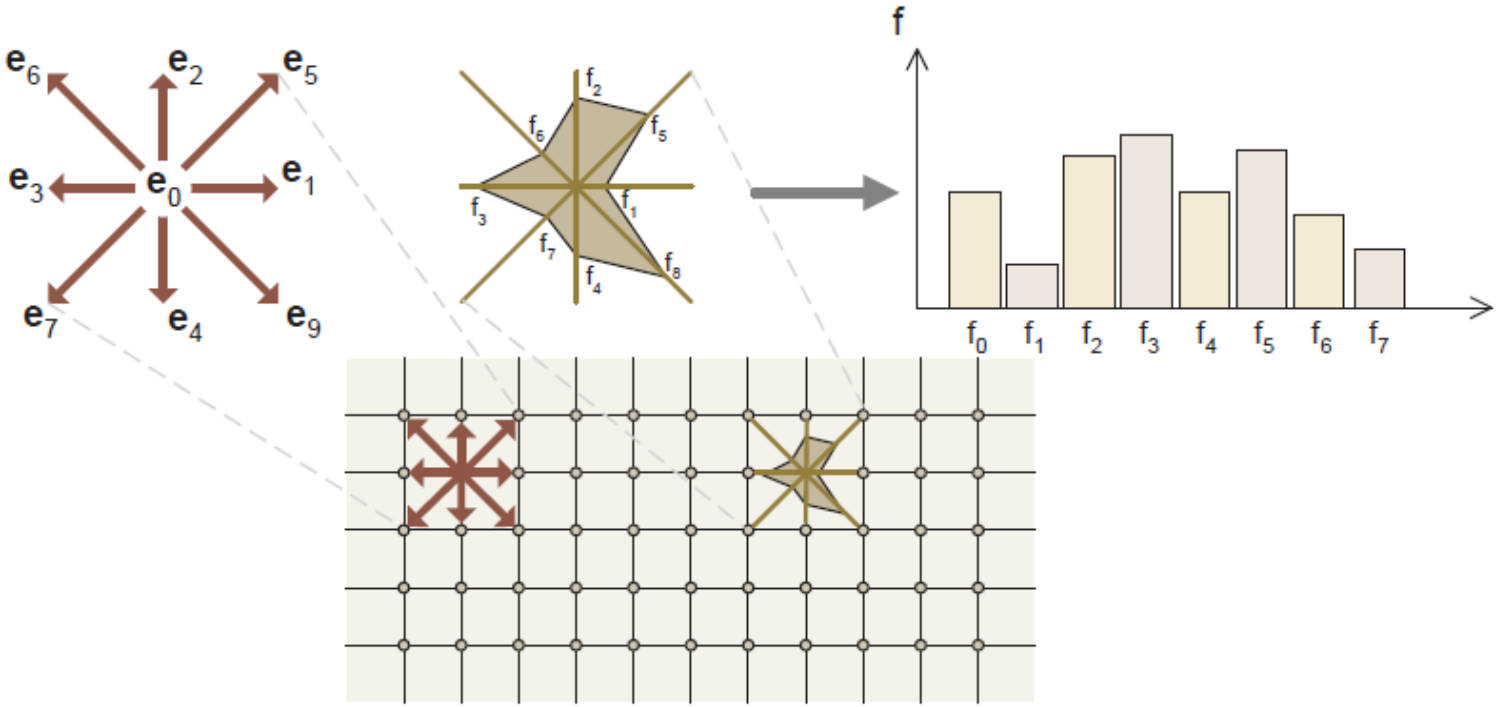
- Historycznie wprowadzona jako rozwinięcie LGA



- Zmienne n_i (0,1) zastąpione funkcją rozkładu $f_i \in [0, 1]$

Model d2q9

- Dyskretyzacja i model LBM
- Wektory prędkości i funkcja rozkładu na siatce:



$$f \text{ ------} > f_i$$

$$v \text{ ------} > c_i$$

Thorne, D., Sukop, M., *Lattice Boltzmann Method for the Elder Problem*, FIU, CMWR (2004)

Uniwersytet Wrocławski

Równanie transportu (dyskretnie)

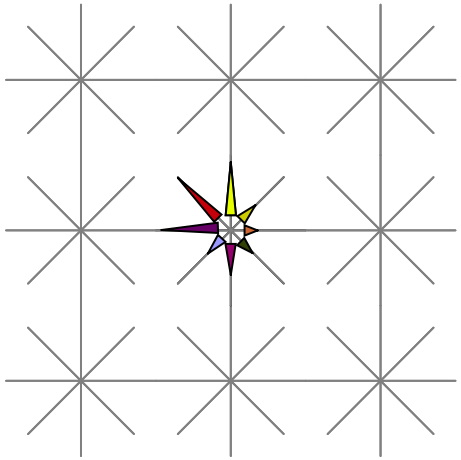
- Równanie transportu w modelu sieciowym Boltzmanna:

$$\underbrace{f_i(\mathbf{x}+\mathbf{c}_i, t+1) - f_i(\mathbf{x}, t)}_{\text{Transport}} = \underbrace{-\frac{1}{\tau}(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t))}_{\text{Collision}},$$

- gdzie zakres *i* zależy od użytej sieci.

Uniwersytet Wrocławski

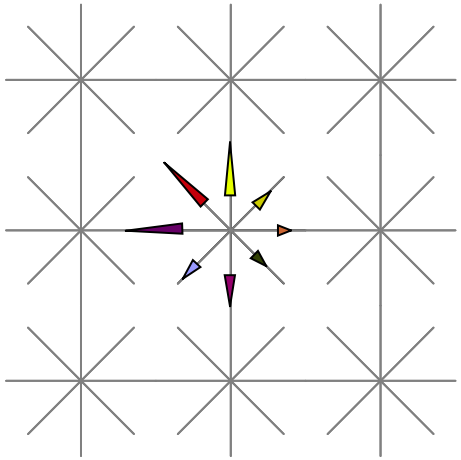
Krok transportu



- $t=0$

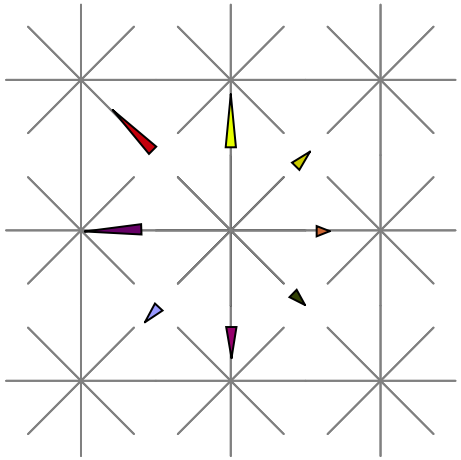
Uniwersytet Wrocławski

Krok transportu



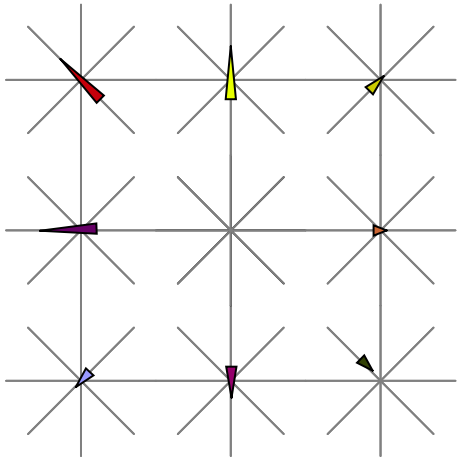
- $t=1/4$

Krok transportu



- $t=1/2$

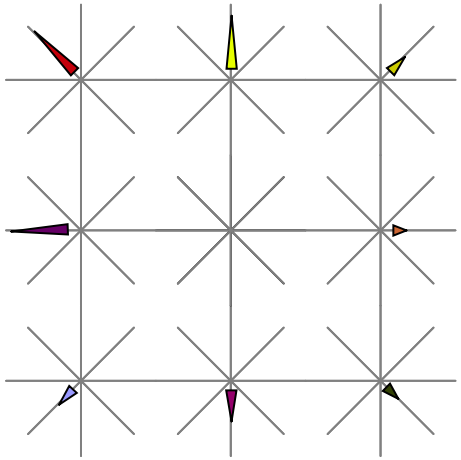
Krok transportu



- $t=3/4$

Uniwersytet Wrocławski

Krok transportu



- $t=1$

Uniwersytet Wrocławski

Podsumowanie: Algorytm LBM

Podstawowy algorytm **LBM** to dwa kroki:

- kolizji:

$$\tilde{f}_i(\mathbf{x}, t) = f_i(\mathbf{x}, t) - \frac{1}{\tau}(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t))$$

- transportu:

$$f_i(\mathbf{x} + \mathbf{c}_i, t + 1) = \tilde{f}_i(\mathbf{x}, t).$$

http://www.ift.uni.wroc.pl/~koma/pr/cudamm/lbm-mini.zip

Podstawowy algorytm LBM

- Można powiedzieć, że poniższy kod to pełny algorytm LBM (podstawowy!)

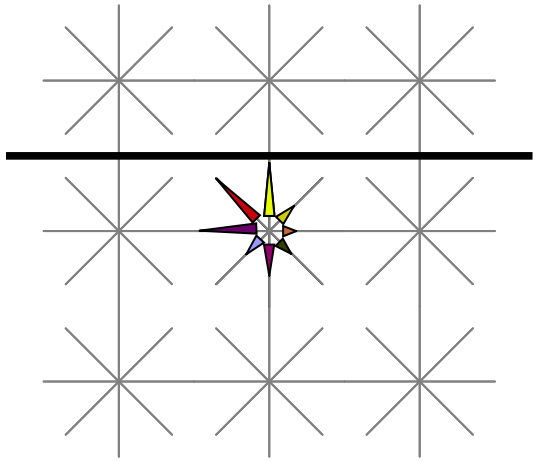
```
for(int k=0; k< 9; k++)
{
        int ip = ( i+ex[k] + L ) % (L);
        int jp = ( j+ey[k] + L ) % (L);
        // collision + streaming

        if( FLAG[ip+jp*L] == 1 )
                df [1-c][ i+j*L ][inv[k]] = (1-omega) *
                df[c][ i+j*L ][k] + omega* w[k] * rho *
                (1.0f - (3.0f/2.0f) * (ux*ux + uy*uy) +
                3.0f * (ex[k] * ux + ey[k]*uy) +
                (9.0f/2.0f) * (ex[k] * ux + ey[k]*uy) *
                (ex[k] * ux + ey[k]*uy));
        else
                df [1-c][ip+jp*L][k] = (1-omega) * df[c][
                i+j*L ][k] + omega* w[k] * rho * (1.0f -
                (3.0f/2.0f) * (ux*ux + uy*uy)+ 3.0f * (ex[k] * ux
                + ey[k]*uy) + (9.0f/2.0f) * (ex[k] * ux +
                ey[k]*uy) * (ex[k] * ux + ey[k]*uy));
}
```

Pełny kod w matlabie na 1 stronie: http://www.exolete.com/code/lbm

Uniwersytet Wrocławski

Warunki brzegowe

- Bounce-back na sztywnej ściance

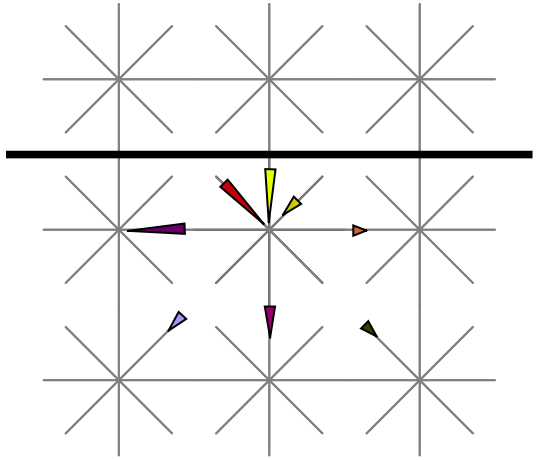


- Można uzyskać dokładność drugiego rzędu stosując wersję „mid-grid”

Succi, Sauro (2001). The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press

Warunki brzegowe

- Bounce-back na sztywnej ściance



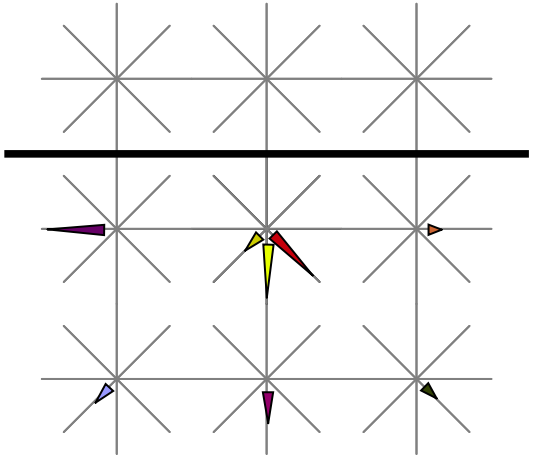
- Można uzyskać dokładność drugiego rzędu stosując wersję „mid-grid”

Succi, Sauro (2001). The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press

Uniwersytet Wrocławski

Warunki brzegowe

- Bounce-back na sztywnej ściance



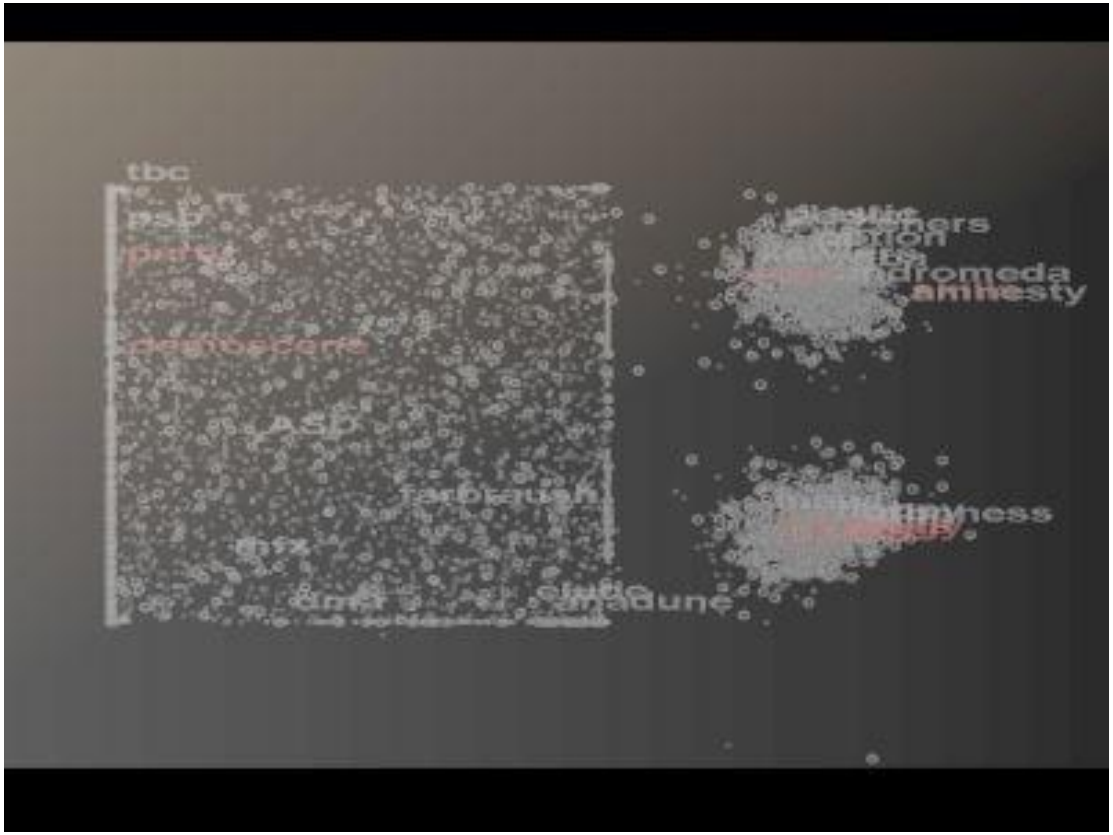
- Można uzyskać dokładność drugiego rzędu stosując wersję „mid-grid”

Succi, Sauro (2001). The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press

Uniwersytet Wrocławski

Pantha Rhei / Floppy

- Demoscena
- 4kb intro (4096 bajtów)
- Animacja czasu rzeczywistego (LBM)



Uniwersytet Wrocławski

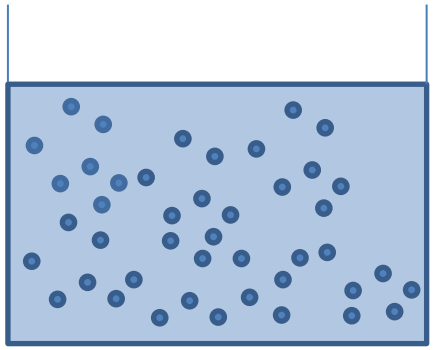
Metody rozwiązania

Skala makro (continuum)

- Navier–Stokes Equations (NSE)
  - Finite Difference Method (FDM)
  - Finite Volume Method (FVM)
  - Finite Element Method (FEM)
- **Smoothed Particle Hydrodynamics** (SPH)
- Dissipative Particle Dynamics (DPD)
- The Lattice Boltzmann Method (LBM)
- Lattice Gas Automata (LGA)
- Molecular Dynamics (MD)

Skala mikro (poziom atomowy)

Hydrodynamika cząstek rozmytych



1. Lagrangian Fluid Dynamics, Using Smoothed Particle Hydrodynamics, Micky Kelager, 2006

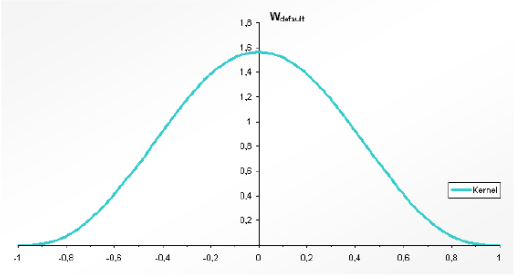
2. http://pl.wikipedia.org/wiki/SPH)

- 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.

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

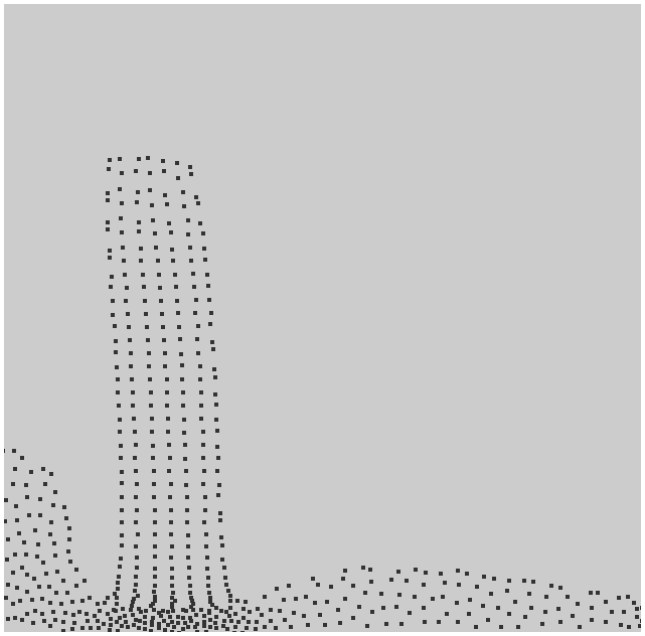


Algotytm SPH

- Inicjalizuj pozycje i prędkości cząstek SPH

- Oblicz gęstość cząstki (z kerneli rozmycia)
- Oblicz siły działające na cząstkę
  - ciśnienie
  - lepkość,
  - siły powierzchniowe
- Przesuń cząstki
  (najprościej schemat Euler…)

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



(SPH w działaniu)

Uniwersytet Wrocławski

Numb res / Fairlight

- Demoscena
- demo (najważniejsza kategoria)
- Animacja czasu rzeczywistego (SPH, shadery GPU)
- 2 miejsce na The Gathering 2011

Uniwersytet Wrocławski

Koniec

Dziękuję za uwagę, więcej?

www.matyka.pl