Fizyka w modelowaniu i symulacjach komputerowych
Jacek Matulewski (e-mail: jacek@fizyka.umk.pl)
http://www.fizyka.umk.pl/~jacek/dydaktyka/

Symulacje komputerowe

Zbiór punktów materialnych

Wersja: 18 lutego 2010

Plan

1. Model punktu materialnego. Kinematyka
2. Dynamika punktu materialnego (© I. Newton)
3. Modelowanie oddziaływań punktów materialnych. Równania ruchu
4. Obszar niedostępny. Siły kontaktowe.
   Odbicie kuli od nieruchomej powierzchni
5. Zderzenie niecentralne i niesprężyste dwóch kul
   a) detekcja kolizji (geometria)
   b) reakcja na zderzenie (fizyka)

Koncepcja punktu materialnego

- Rozmiar ciała jest nieistotny w porównaniu z innymi odległościami zagadnienia (np. z przebytą drogą)
- Masa ciała skupiona jest w punkcie geometrycznym (najlepiej w środku masy)
- Ruch postępowy, bez obrotów!
- Implementacja w klasach PunktMaterialny i ZbiorPM

Równania ruchu

- Druga zasada dynamiki

$$\vec{F}(\vec{r},\vec{v},t) = m\vec{a}(t)$$

- Równanie na położenie punktu (konkretna siła) np. w pobliżu powierzchni Ziemi

$$m\vec{a}(t) = m\vec{g}$$

- Rozwiązanie wymaga warunków początkowych:

$$\vec{r}(t_0 + \Delta t) = \vec{r}(t_0) + \vec{v}(t_0)\Delta t + \frac{\vec{g}\Delta t^2}{2}$$

Metoda Eulera

- Taylor 1-go rzędu (jak ilorazy różnicowe)
- Przepis:
  - Oblicz prędkość w kolejnej chwili czasu:

    $$\vec{v}(t+\Delta t) \approx \vec{v}(t) + \vec{a}(t)\Delta t$$

  - Oblicz położenie w kolejnej chwili korzystając z pr.

    $$\vec{r}(t+\Delta t) \approx \vec{r}(t) + \vec{v}(t+\Delta t)\Delta t$$

- Zależy od położenia i prędkości w jednej chwili czasu (wygodne przy zmianie stanu punktu)

Metoda Verleta

- Przepis:
  - Obliczani położenia w następnej chwili czasu:

$$\vec{r}(t+\Delta t) \approx -\vec{r}(t-\Delta t) + 2\vec{r}(t) + \vec{a}(t)\Delta t^2$$

  - Do kolejnych kroków i do wizualizacji prędkość wcale nie jest potrzebna (można jej nie obliczać)!

$$\vec{v}(t) \approx \frac{\vec{r}(t+\Delta t) - \vec{r}(t-\Delta t)}{2\Delta t}$$

- Większa dokładność przy tej samej złożoności obliczeniowej. Metoda dynamiki molekularnej

Metoda Eulera i Verleta

Implementacja w klasie PunktMaterialny

```
template<typename T>
void TPunktMaterialny<T>::PrzygotujRuch_Euler(TWektor<T> przyspieszenie,T krokCzasowy)
{
        nastepnaPredkosc=predkosc+przyspieszenie*krokCzasowy;
        nastepnePolozenie=polozenie+nastepnaPredkosc*krokCzasowy;
}

#define SQR(x) ((x)*(x))

template<typename T>
void TPunktMaterialny<T>::PrzygotujRuch_Verlet(TWektor<T> przyspieszenie,T krokCzasowy)
{
        if(numerKroku==0) PrzygotujRuch_Euler(przyspieszenie,krokCzasowy);
        else
        {
                nastepnePolozenie=2.0*polozenie-poprzedniePolozenie+przyspieszenie*SQR(krokCzasowy);
                nastepnaPredkosc=(nastepnePolozenie-poprzedniePolozenie)/(2*krokCzasowy);
        }
}
```

Metoda Eulera i Verleta

Implementacja w klasie ZbiorPunktowMaterialnych

```
template<typename T>
void TZbiorPunktowMaterialnych<T>::PrzygotujRuch(T krokCzasowy,Algorytm algorytm)
{
        for(int i=0;i<ilosc;++i)
                if(!wiezy[i])
                        this->punkty[i].PrzygotujRuch(Sila(i),krokCzasowy,algorytm);
}

template<typename T>
void TZbiorPunktowMaterialnych<T>::WykonajRuch()
{
        for(int i=0;i<ilosc;++i)
                if(!wiezy[i])
                        this->punkty[i].WykonajRuch();
}

template<typename T>
void TZbiorPunktowMaterialnych<T>::KrokNaprzod(T krokCzasowy,Algorytm algorytm)
{
        PrzedKrokiemNaprzod(krokCzasowy);
        PrzygotujRuch(krokCzasowy,algorytm);
        PoPrzygotowaniuRuchu(krokCzasowy);
        WykonajRuch();
        PoKrokuNaprzod(krokCzasowy);
}
```

Siły sprężystości

- Siła harmoniczna - oscylacje
- Siła tłumiąca
- Siła oporu

Siła harmoniczna

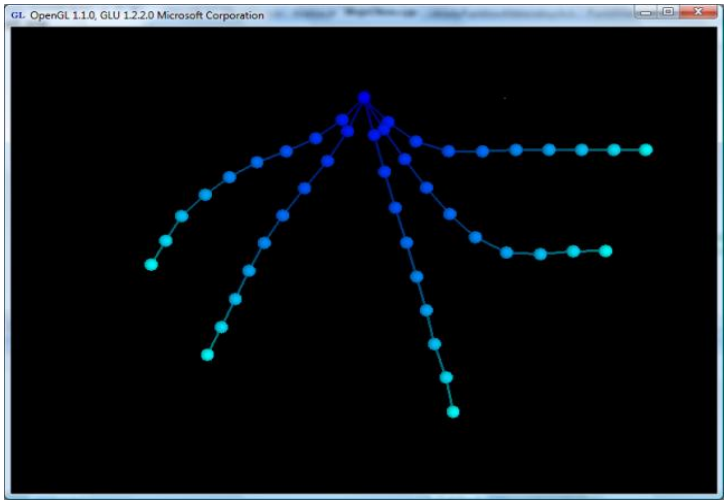
$$\vec{F}(t) = -k\Delta\vec{r} - 2\beta_t(\Delta\vec{v}\cdot\Delta\hat{r})\Delta\hat{r} - 2\beta_o\vec{v}$$

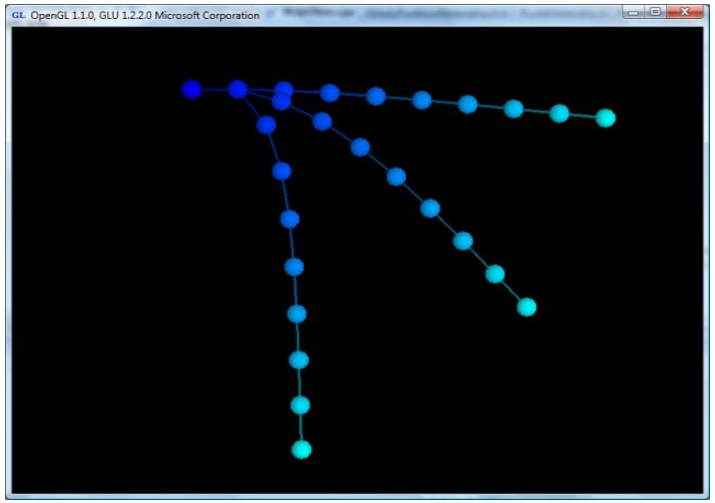
tłumienie oscylacji opory

Siły sprężystości

- Oscylatory sprzężone (demonstracja)

- Sztywność (siła zależąca od kąta vs. dalsi sąsiedzi
- Lina i włos (demonstracja)





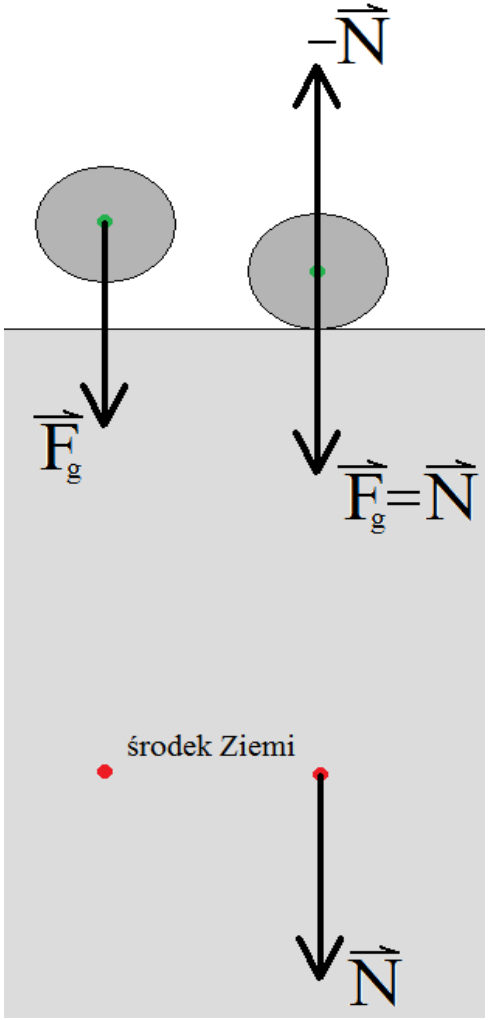
Siły kontaktowe

- Ciało nieruchome (masa, więzy), obszar niedostępny (zabroniony), nieruchomy obszar kolizji
- Równoważenie składowej siły normalnej do powierzchni

$$\vec{F}_{\parallel} = \vec{F} - (\vec{n} \cdot \vec{F})\vec{n} \text{ , gdy } \vec{n} \cdot \vec{F} < 0$$

- Modyfikacja prędkości (odbicie)

$$\vec{v}_{\parallel} = \vec{v} - (w_{odb} + 1)(\vec{n} \cdot \vec{v})\vec{n}$$



Siły kontaktowe

- Siła tarcia
  (składowa równoległa siły kontaktowej)

$$\vec{T} = -\mu \left| \vec{N} \right| \frac{\vec{v}}{\left| \vec{v} \right|}$$

- Tarcie nie zależy od rozmiaru powierzchni styku (Leonardo da Vinci, Amontos)
- Tarcie dynamiczne i statyczne

Siły kontaktowe

- Implementacja – klasa ObszarZabroniony
- Pole klasy ZbiorPunktowMaterialnych

Zadania:

- Czy punkt wszedł do obszaru niedostępnego?
- Określenie normalnej do powierzchni w punkcie penetracji

Siły kontaktowe

Implementacja – klasa ObszarZabroniony

```
template<typename T> class TObszarZabroniony
{
	private:
		T wspolczynnikOdbicia;
		T wspolczynnikTarcia;
	public:
		TObszarZabroniony(T wspolczynnikOdbicia,T wspolczynnikTarcia);
		T PobierzWspolczynnikOdbicia();
		T PobierzWspolczynnikTarcia();
		virtual bool CzyWObszarzeZabronionym(TWektor<T> polozenie,
                                                     TWektor<T> poprzedniePolozenie,
                                                     T margines,
                                                     TWektor<T>* normalna) = 0;
};
```

Zaimplementowane obszary zabr.

Podłoże

```
bool CzyWObszarzeZabronionym(Wektor polozenie,
                                        Wektor poprzedniePolozenie,
                                        double margines,Wektor* normalna)
{
        bool wynik=(polozenie.Y<poziomY+margines);
        if(wynik && normalna!=NULL) *normalna=Wektor(0,1,0);
        return wynik;
}
```

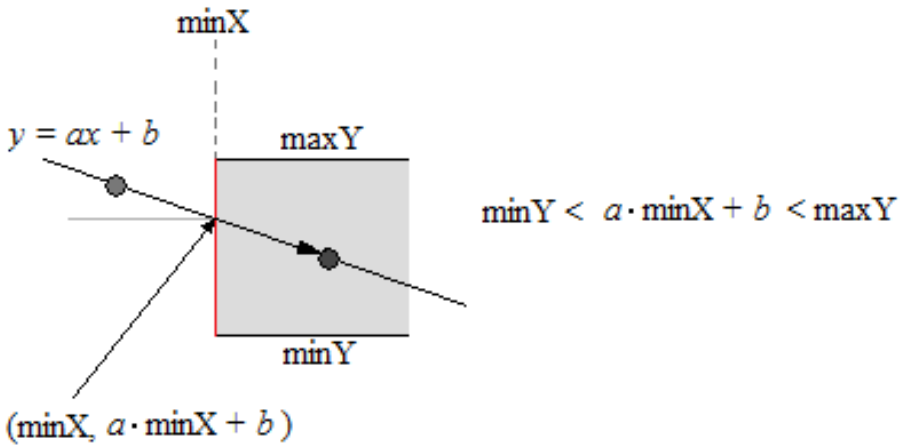
Zaimplementowane obszary zabr.

Kula

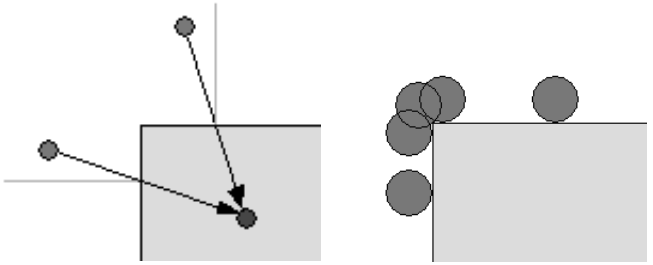
```
bool CzyWObszarzeZabronionym(Wektor polozenie,
                                        Wektor poprzedniePolozenie,
                                        double margines,Wektor* normalna)
{
        Wektor wektorPromienia=polozenie-srodek;
        bool wynik=wektorPromienia.Dlugosc()<promien+margines;
        if(wynik && normalna!=NULL)
        {
                *normalna=wektorPromienia;
                normalna->Normuj();
        }
        return wynik;
}
```

Zaimplementowane obszary zabr.

- Podłoże, kula, walec (demonstracje)
- Walec nieograniczony w kierunku OZ (demo.)
- Prostopadłościan nieograniczony w kierunku OZ

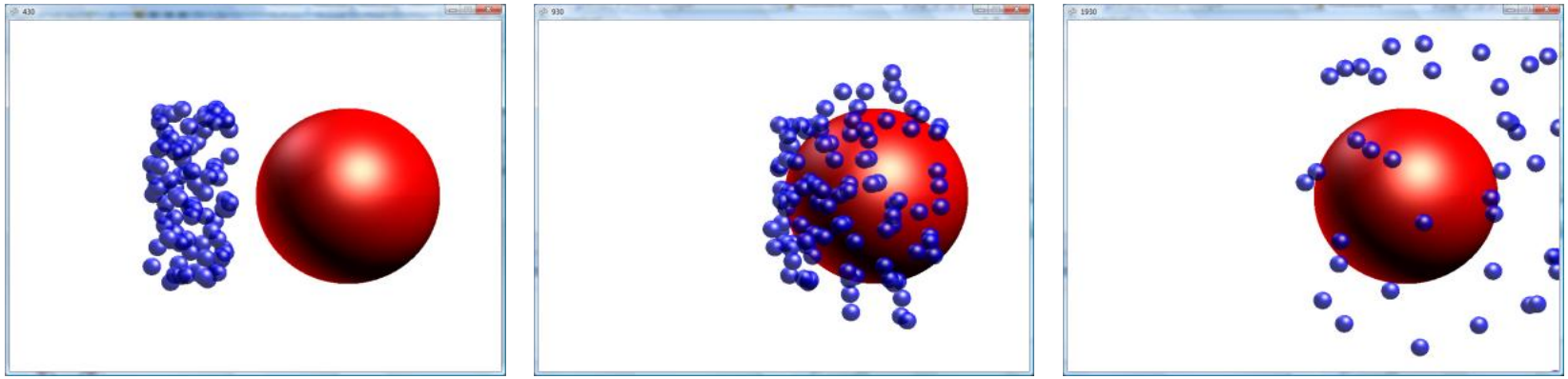


- Uwzględnienie marginesu (rozmiar punktu mat.)



Rozpraszanie na kuli

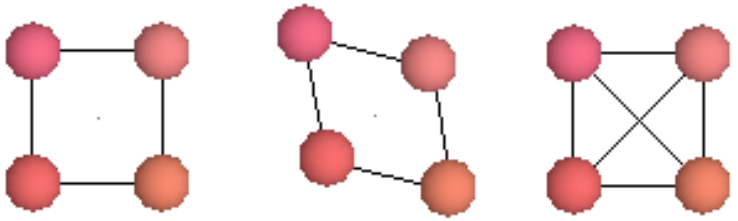
- Demonstracja
- Woda (brak odbicia, $w_{obs}=0$)

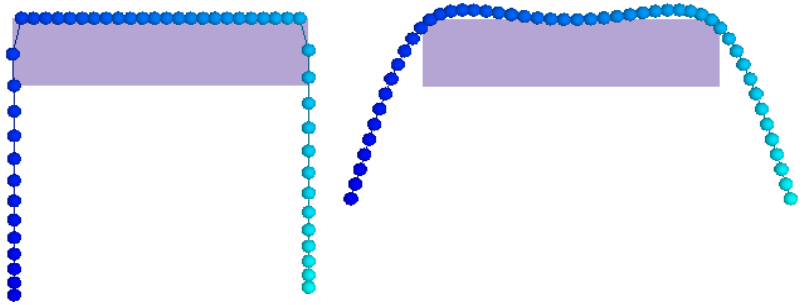


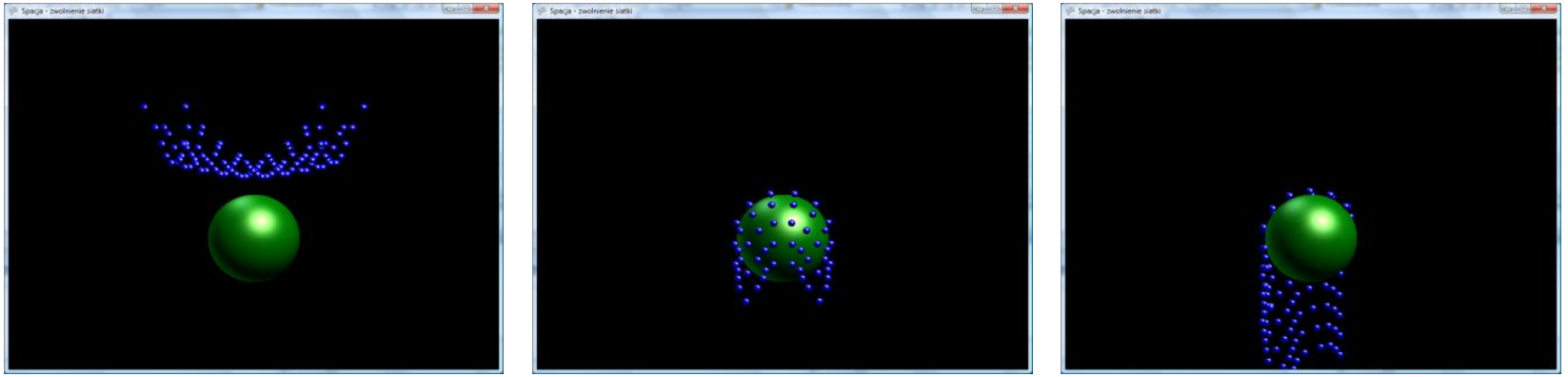
- Metalowe kulki (odbicie sprężyste , $w_{obs}=1$)

Układy dwuwymiarowe (np. tkaniny)

- Ile potrzeba wiązań?

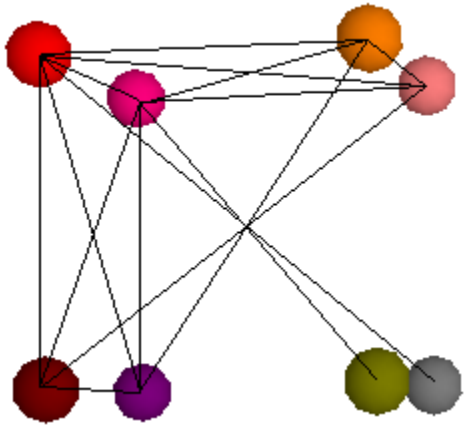


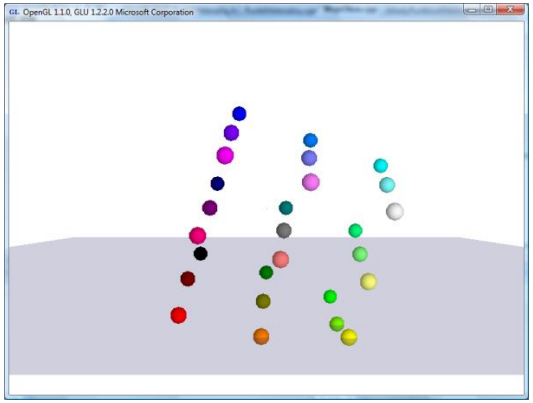


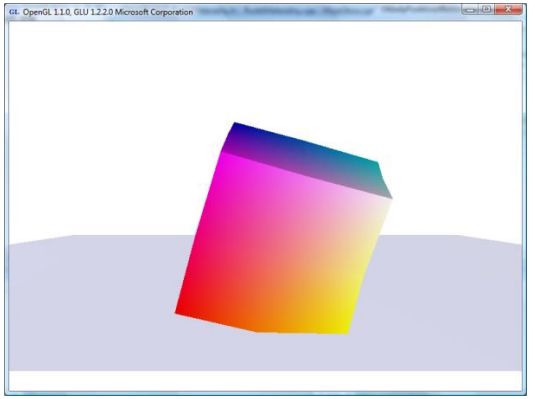


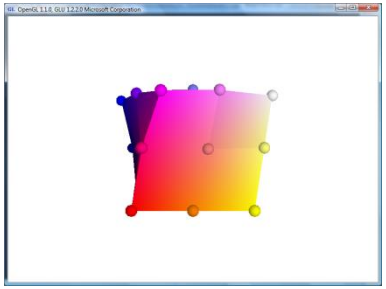
Układy trójwymiarowe (c. miękkie)

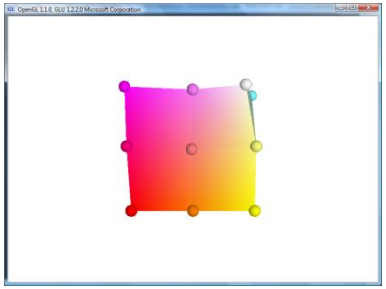
- Ile wiązań?

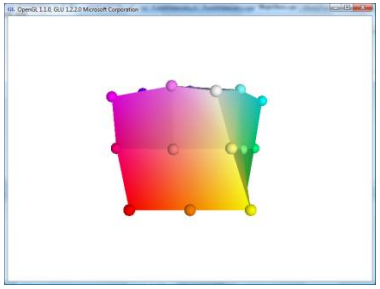












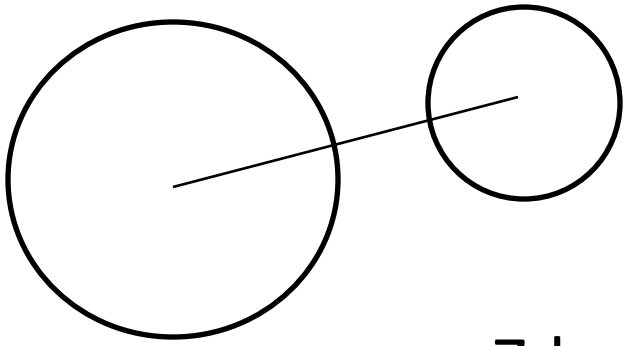
Zderzenia

Dwa etapy:

- Detekcja kolizji (geometria zderzenia)
- Reakcja na zderzenie (fizyka zderzenia)

Niesprężyste i niecentralne zderzenie dwóch kul

Detekcja kolizji (geometria zderzenia)

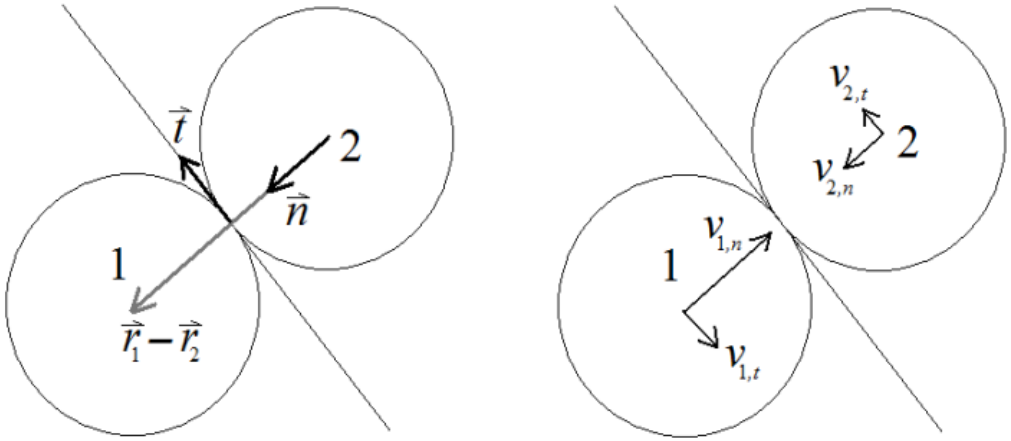


Zderzenie jeżeli odległość środków jest nie większa od sumy promieni

Niesprężyste i niecentralne zderzenie dwóch kul

Reakcja na kolizję (fizyka zderzenia)

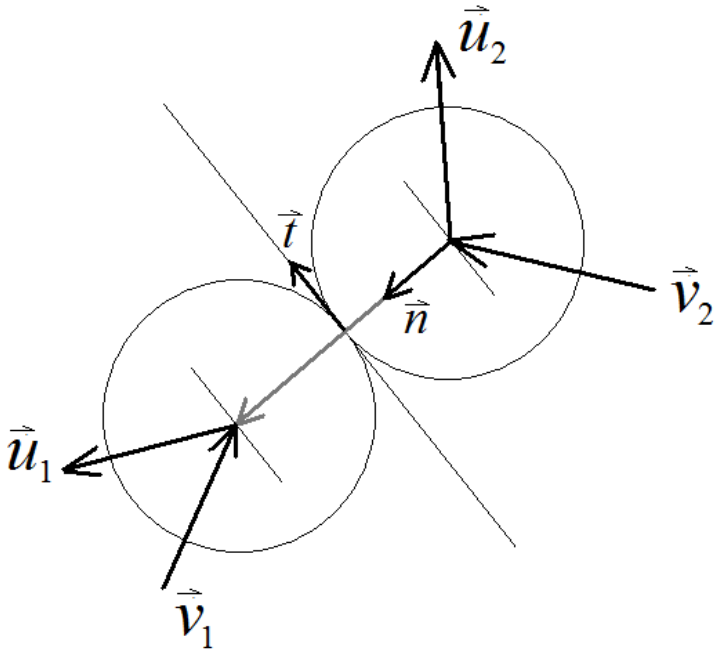
Zagadnienie jest w istocie jednowymiarowe



Normalna zderzenia: $\vec{n} = \left(\vec{r}_2 - \vec{r}_1\right) \Big/ \left|\vec{r}_2 - \vec{r}_1\right|$

Niesprężyste i niecentralne zderzenie dwóch kul

- Popęd = zmiana pędu w momencie zderzenia
- Jeżeli nie ma tarcia, nie ma obrotów: $\vec{J} = J\,\vec{n}$



$$J = m_1\left(u_{1n} - v_{1n}\right)$$

$$-J = m_2\left(u_{2n} - v_{2n}\right)$$

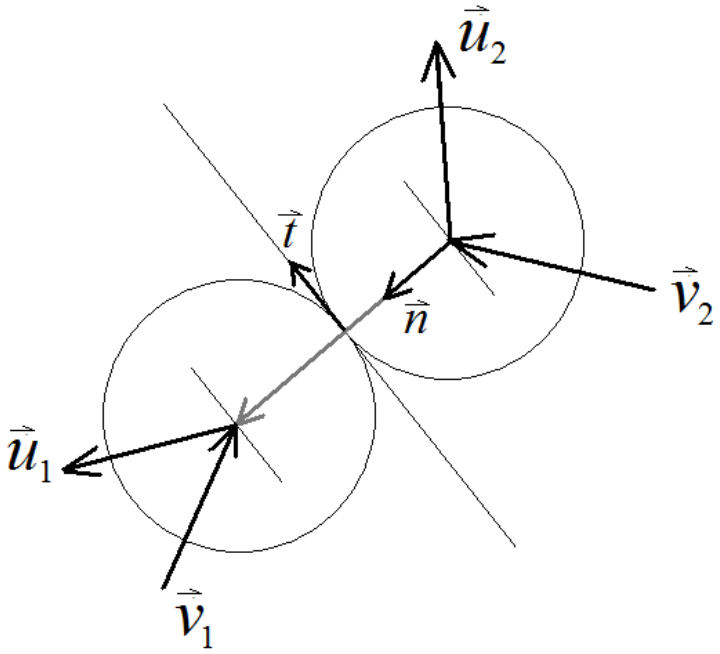
$$e = -\frac{u_{1n} - u_{2n}}{v_{1n} - v_{2n}}$$

Trzy niewiadome!!!

współczynnik restytucji
(miara niesprężystości)

Niesprężyste i niecentralne zderzenie dwóch kul

- Popęd = zmiana pędu w momencie zderzenia
- Jeżeli nie ma tarcia, nie ma obrotów $\vec{J} = J\,\vec{n}$



Rozwiązanie:

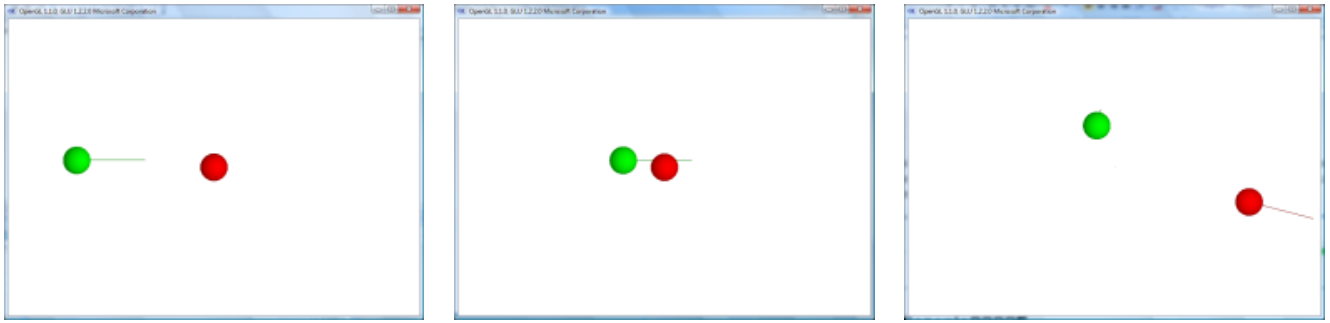
$$J = -\mu(e+1)\left(v_{1n} - v_{2n}\right)$$

$$\vec{u}_1 = \vec{v}_1 + \frac{J}{m_1}\vec{n}$$

$$\vec{u}_2 = \vec{v}_2 - \frac{J}{m_2}\vec{n}$$

Niesprężyste i niecentralne zderzenie dwóch kul

- Test – problem bilardzisty (demostracja)
  Jeżeli bile mają równe masy, to powinny rozbiec się pod kątem prostym.



- Bilard (demonstracja)

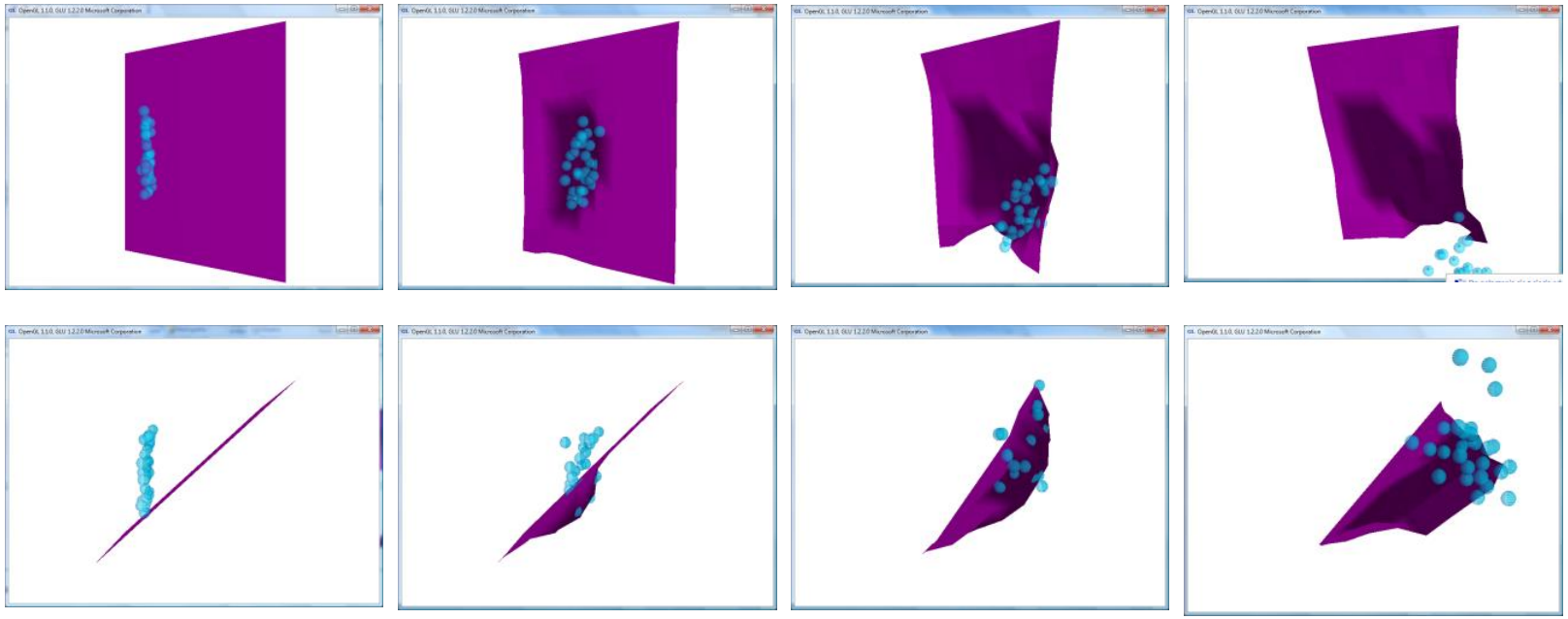
Sterowanie – „fizyczna” ingerencja

- Zmiana pozycji punktu myszą – sprężyna między położeniem punktu i myszy w 3D
- Bezpośrednie związanie – problemy numeryczne
- Demonstracja

- Zmiana pozycji za pomocą klawiatury – kontrola prędkości czy przyspieszenia
- Demonstracja

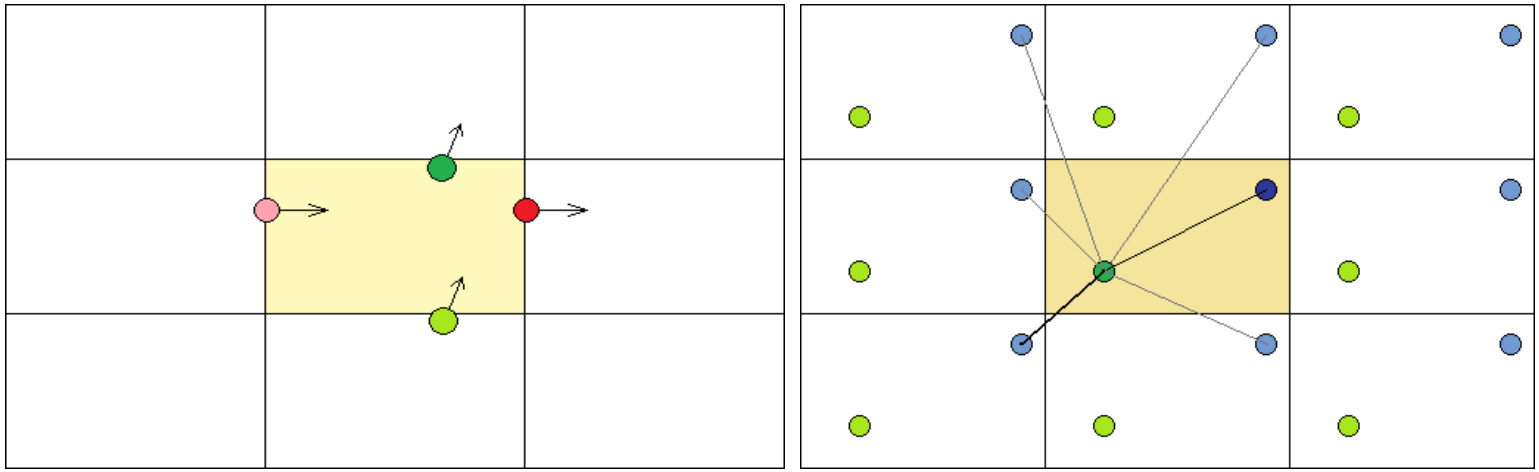
Woda uderzająca w tkaninę

- Demostracja – źródła niedoskonałości
- Przecięcie odcinka z trójkątem
- Omówienie implementacji



Optymalizacja przy dużej ilości pkt.

- Periodyczne warunku brzegowe (PBC)



- Podział na komórki – ograniczenie ilości oddział.
- Złożoność oblicz. problemu z N^2 zmienia się w N
- Analiza implementacji

Przykład użycia implementacji zbioru punktów materialnych

World of Goo (demonstracja)