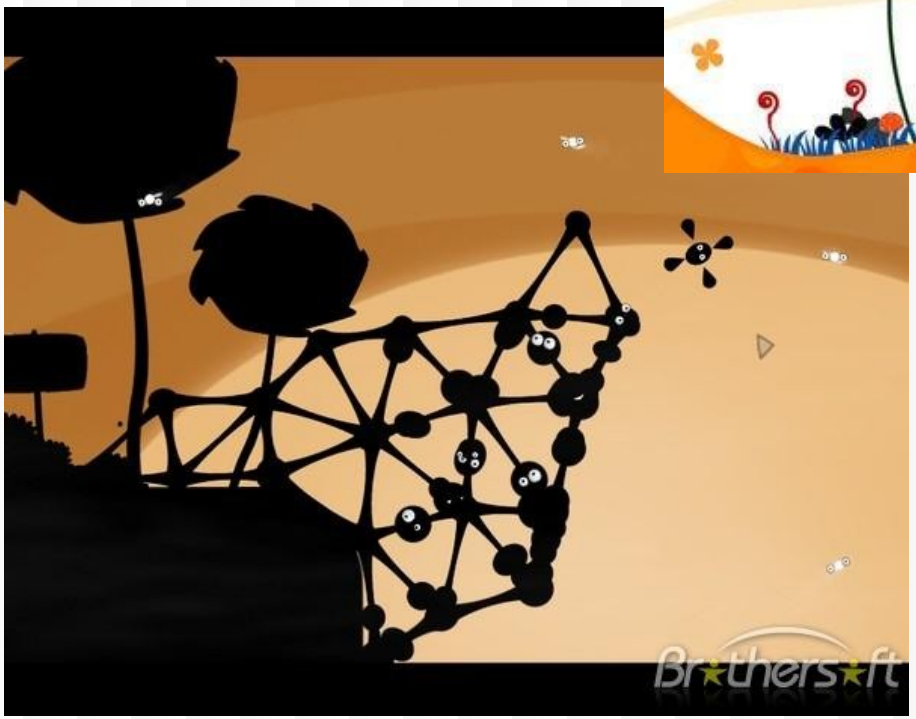
Symulacja Liny
Metoda Verleta

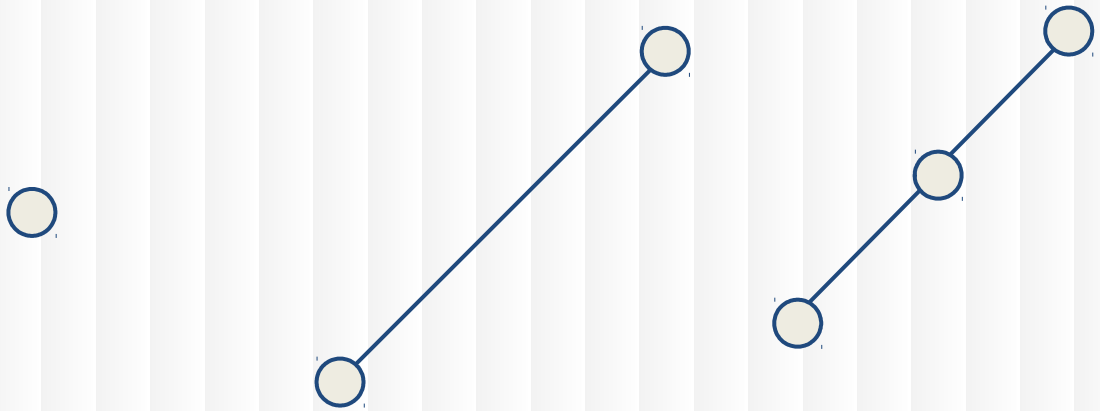
Modelowanie fizyczne w animacji komputerowej

Maciej Matyka
http://panoramix.ift.uni.wroc.pl/~maq/

Fizyka czasem się przydaje



O czym będzie ten wykład?



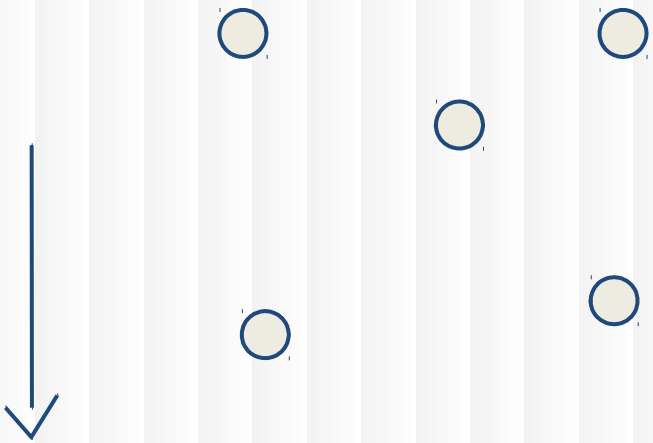
Jak zbudować układy mas połączonych sprężynkami?

Jak je opisać w programie?

Jak wyznaczyć siły, a następnie ruch?

Co ciekawego można z tego zbudować?

Punkty w polu grawitacyjnym



Punkt materialny

Punkt materialny cechują:

- masa (kg)
- pozycja (m)
- prędkość (m/s)
- siła (kg*m/s^2)

...

```
struct point
{
    float m;
    CVector3 r;
    CVector3 v;
    CVector3 f;
};
```

Prosty przepis

1. Inicjalizuj punkty
2. Policz siły
3. Przesuń punkty
4. Więzy, kolizje, korekty, odbicia..
5. Wróć do 2.

Prosty przepis

1. **Inicjalizuj punkty**
2. Policz siły
3. Przesuń punkty
4. Więzy, kolizje..
5. Wróć do 2.

```
void initpoints(void)
{

    for(int i=0; i<N; i++)
    {

        float r1 = rand()/float(RAND_MAX);
        float r2 = rand()/float(RAND_MAX);
        float r3 = rand()/float(RAND_MAX);
        float r4 = rand()/float(RAND_MAX);

        points[i].v = CVector3( r1 , r2 , 0);
        points[i].r.x = r3;
        points[i].r.y = r4;
        points[i].m = 5;
    }
```

Prosty przepis

1. Inicjalizuj punkty
2. **Policz siły**
3. Przesuń punkty
4. Więzy, kolizje...
5. Wróć do 2.

```
CVector3 g = CVector3(0,-9.8,0);


...

// gravity
for(int i=0; i<N; i++)
{
    points[i].f = g * points[i].m;
}
```

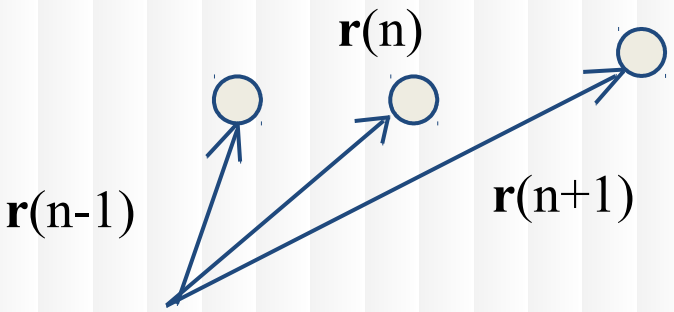
Metoda Verleta

Chcemy przesunąć punkt materialny w nową pozycję **r**(n+1)

Znamy jego starą pozycję **r**(n-1), bieżącą **r**(n) oraz siły **f**(n), które na niego działają.

Lepszym (od meody Eulera) posobem jest wykorzystanie jedno krokowej **metody Verleta**:

$$\mathbf{r}(n+1) = 2*\mathbf{r}(n) - \mathbf{r}(n-1) + h*h*\mathbf{f}(n) / m$$

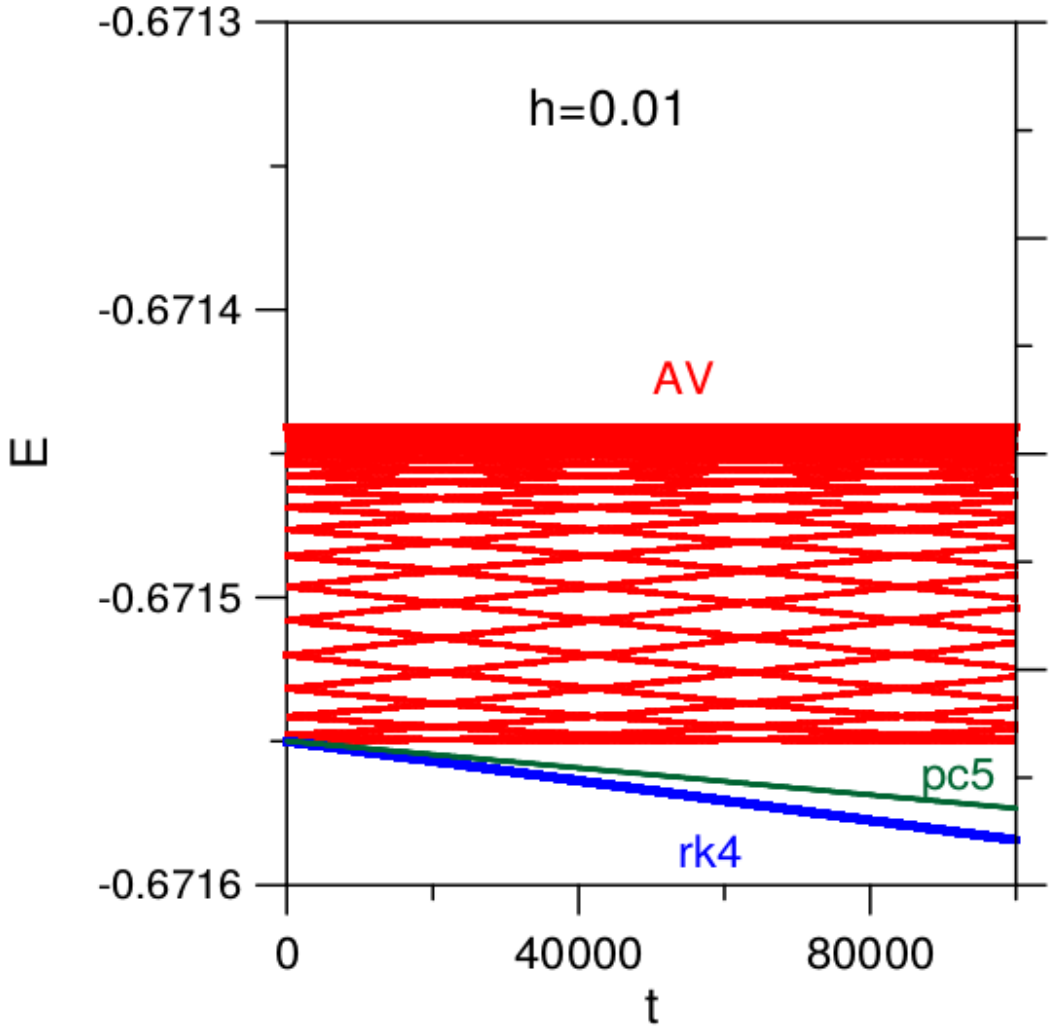


Brakuje tylko początku

Pierwszy krok w metodzie Verleta (np.) :

$$r(1) = r(0) + h*h* \mathbf{f}(0) / m$$

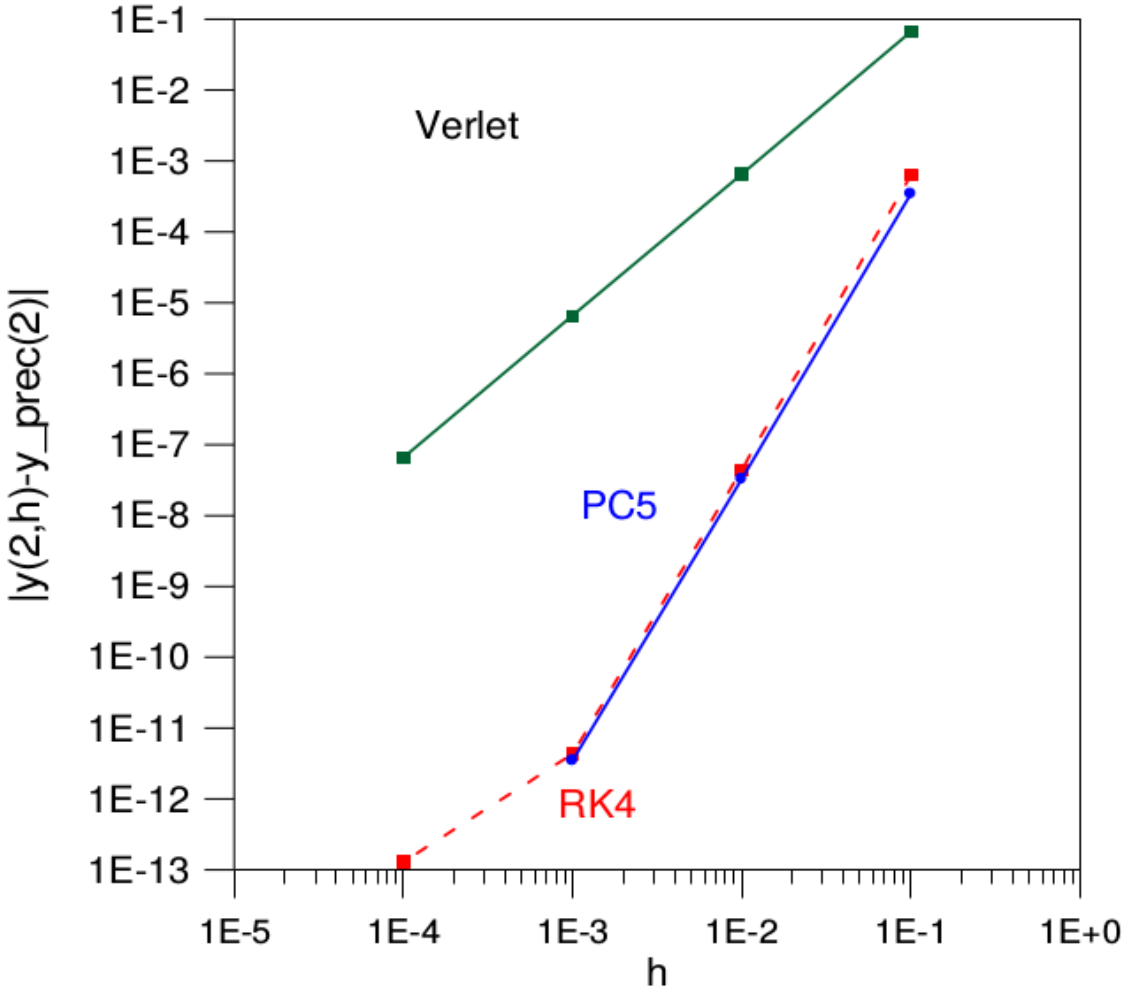
(przykład)



Energia całkowita planety jako funkcja czasu otrzymana trzema metodami z krokiem czasowym h=0.01.

W przypadku algorytmu Verlerta energia jest obliczona z mniejszą dokładnością (czerwony pas na wykresie), ale jej zachowanie jest stabilne. Nie widać dryfu energii.

Energia w metodzie Verlet-a i RK4 (Wykład Cz. Oleksy)



Porównanie dokładności całkowania trzema metodami po upływie czasu t=2. Wynik otrzymany za pomocą PC5 w czasie $t=2$ z $h=1e-4$ jest traktowany jako punkt odniesienia.

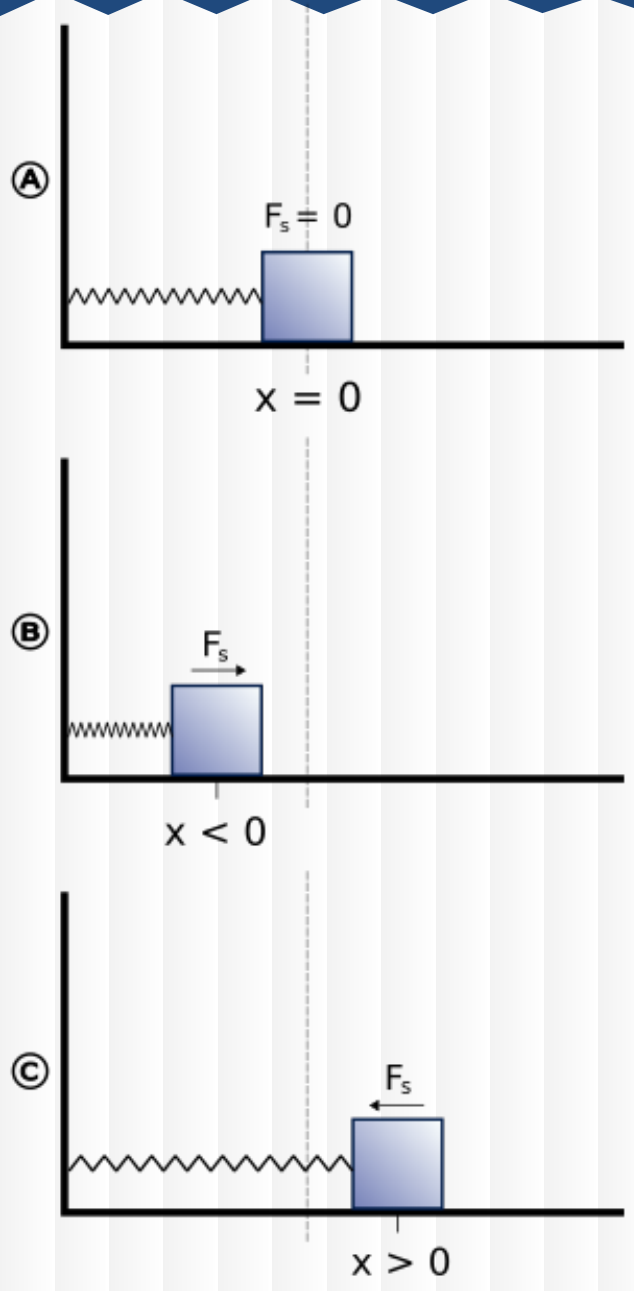
Badanie stabilności tych metod przeprowadziliśmy całkując równania z $h=0.01$ aż upłynął czas $t=10^5$. Energia całkowita powinna być zachowana. Jednak, wyniki otrzymane za pomocą RK4 i PC5 pokazują, że energia wolno, aczkolwiek systematycznie, maleje z czasem.

Dokładność w metodzie Verlet-a, RK4 i Predictor-Corrector 5 (Wykład Cz. Oleksy)

Sprężyna

Sprężyna (prawo Hooke'a)

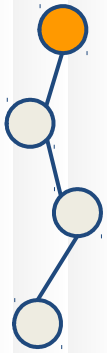
$$F = -kx$$

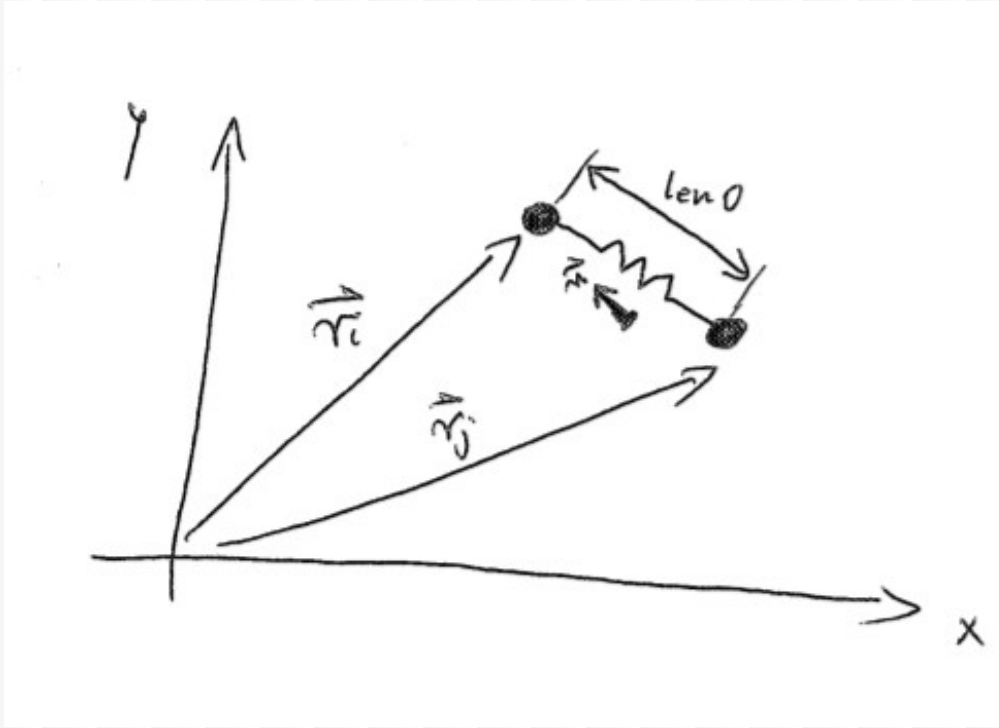


https://en.wikipedia.org/wiki/Harmonic_oscillator

Punkt zawieszony na linie

Siła między punktami: sprężystość + tłumienie

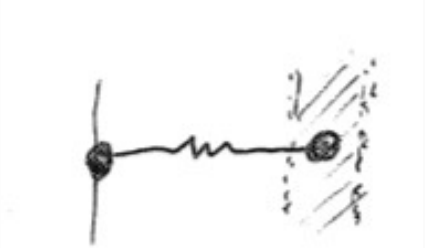




f(i,j) = - ks* (|**ri-rj**| - len0) * **n** + kd * |**v2-v1**|* **n**;

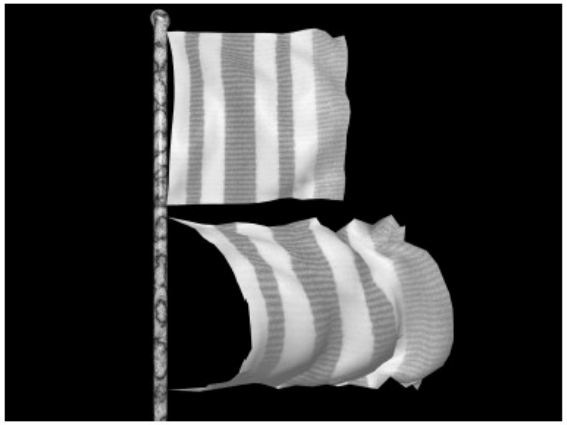
Więzy związane z odległością

Inverse dynamics constraints (Xavier Provot, INRIA):

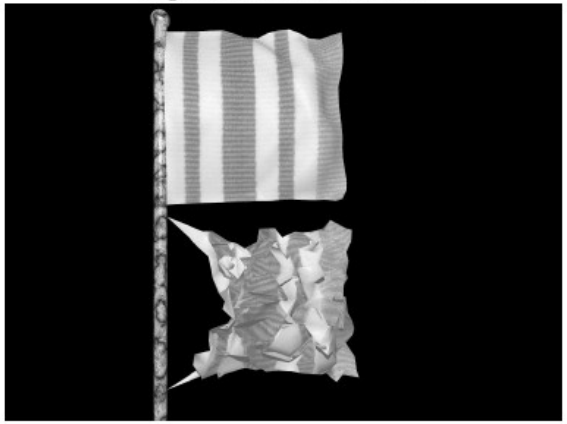


punkty odchylają się tylko o pewien procent od położenia równowagi.

Prosta implementacja - znaczący wzrost stabilności układu

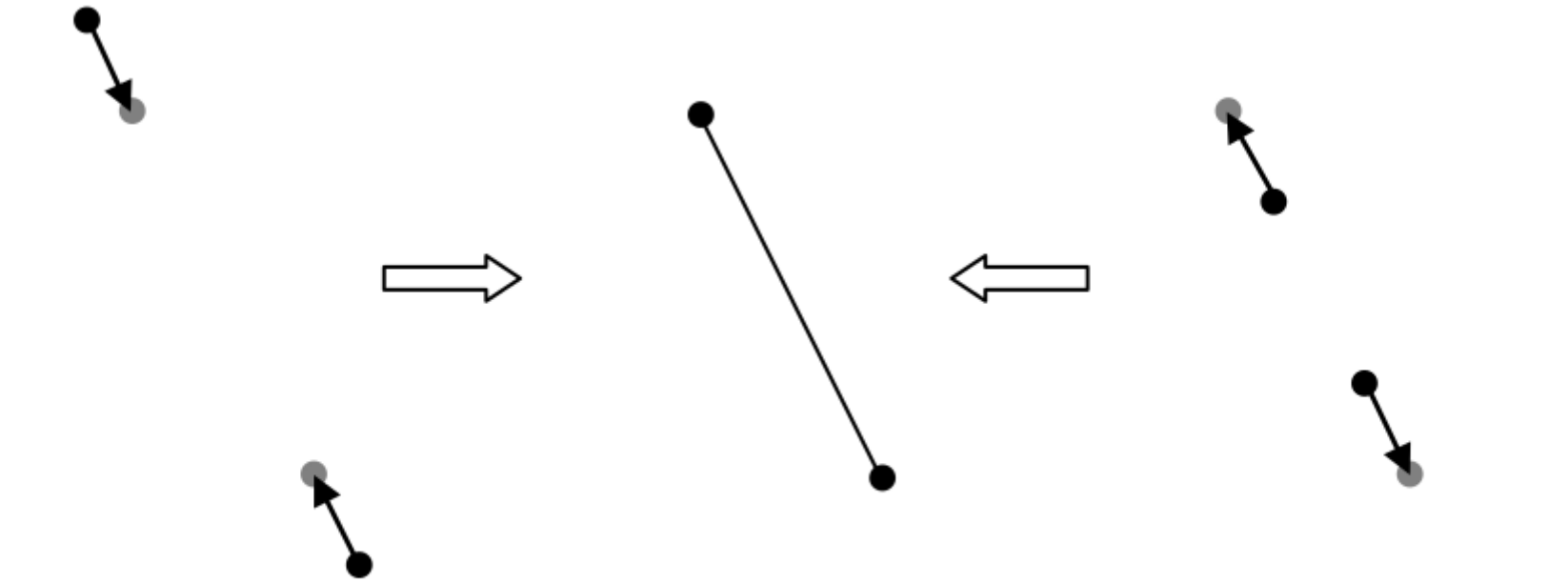


(a) Elastic flag and our "semi-rigid" flag when stiffness is low.



(b) Increasing stiffness: elastic flag becomes chaotic and "semi-rigid" flag remains stable.

Figure 6: Comparison for the case of a flag.


Dist. too large
Correct distance
Dist. too small

HITMEN: Codename 47

HITMEN Codename 47: https://www.youtube.com/watch?v=EWirve3MoUo
Hitman_ Codename 47 Speed Run (0_34_19) - YouTube [360p].mp4
5:44

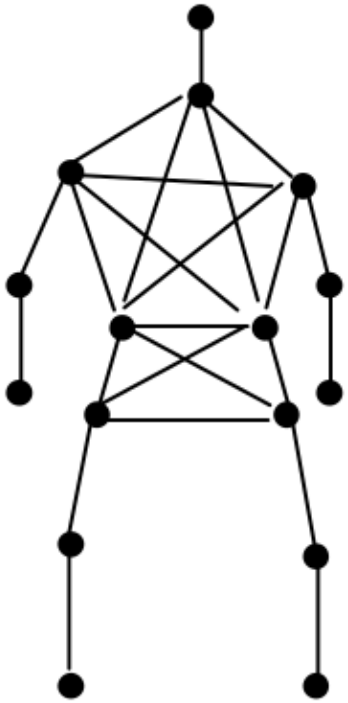


Figure 9. The particle/stick configuration used in Hitman for representing the human anatomy.