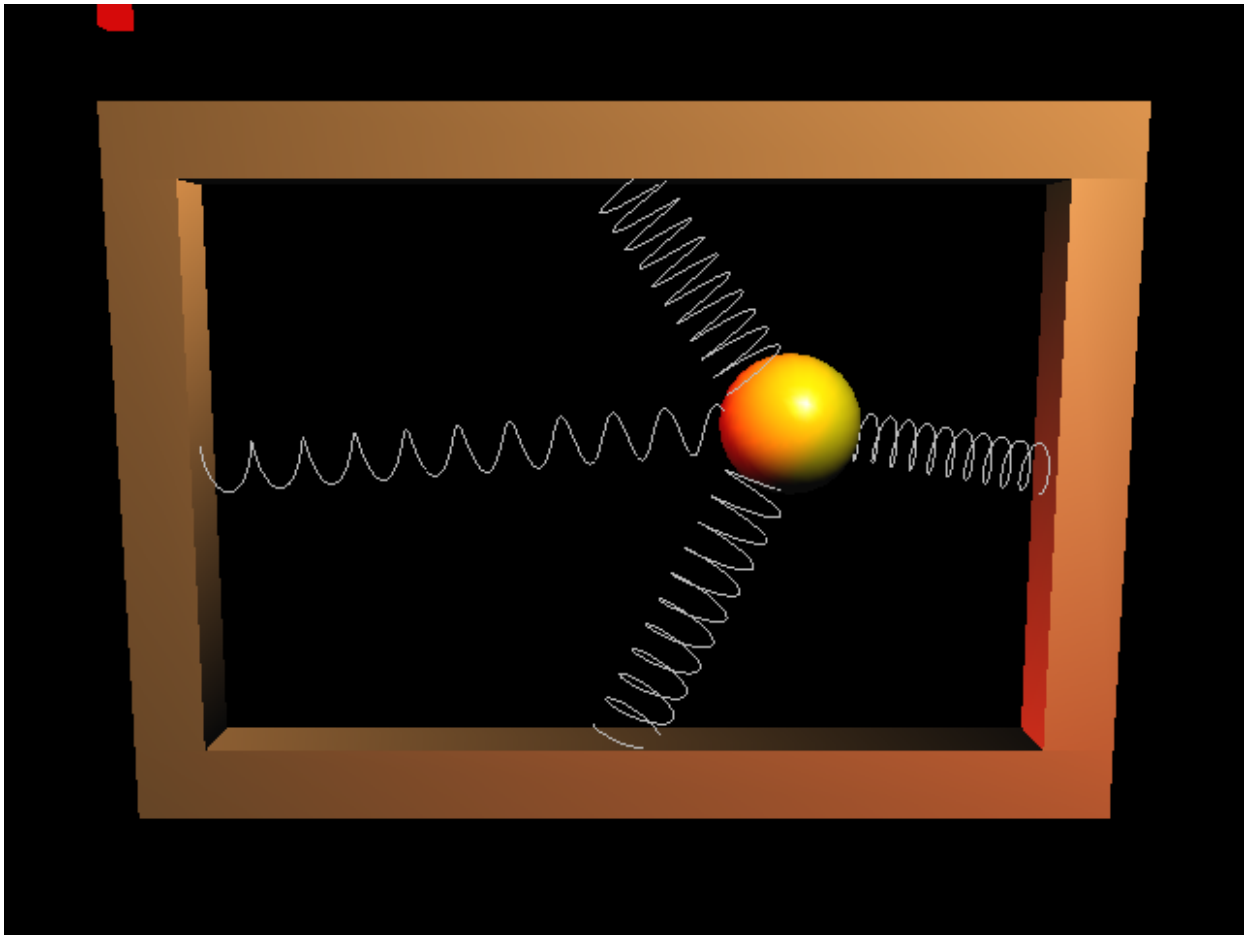


SIMULACIJA FIZIKE

Simulacija fizike

- perspektivnom projekcijom, spremnikom dubine, modelom osvjetljenja i animacijom približili smo se izgledu realnih objekata
- sljedeći korak u realističnosti je da se animirani objekti doista ponašaju kao objekti u stvarnom svijetu
- objekti u stvarnom svijetu ponašaju se u skladu sa zakonima fizike: makroskopski objekti slijede zakone klasične mehanike

Mehanika materijalne točke

- u prvoj aproksimaciji objekt iz stvarnog svijeta aproksimiramo točkom koju opisuju sljedeće fizikalne veličine:
  - **položaj** (koordinate točke)
  - **brzina** (promjena položaja točke u vremenu)
  - **masa** (tromost, mjera opiranja promjeni gibanja)
- ta je aproksimacija u nekim slučajevima opravdana (planeti) ili barem dovoljno dobra za određenu vrstu primjene

Mehanika krutog i deformabilnog tijela

- stvarni objekti nisu točke, već imaju dimenzije
- sljedeća razina opisa uzima u obzir dimenzije objekta, ali te dimenzije tretira kao nepromjenjive – mehanika krutog tijela
- realna tijela se deformiraju – time se bavi mehanika deformabilnog tijela i čitav niz tehničkih disciplina: tehnička mehanika, otpornost materijala, teorija konstrukcija itd.
- tekućinama i plinovima bavi se mehanika fluida

Primjena u 3D grafici

- svaka sljedeća razina sve vjernije opisuje stvarni svijet i može se po potrebi implementirati u aplikacijama koje koriste 3D grafiku
- u praksi (pogotovo u igrama) nije nužno potpuno slijediti zakone fizike (to može biti vrlo zahtjevno za računalne resurse, ali i komplicirano za implementaciju), već se koriste pojednostavljeni modeli pa je onda možda bolje koristiti sintagmu simulacija fizike umjesto fizikalna simulacija

Paradigma fizike

- **bliska je paradigmi objektno-orijentiranog programiranja!**

1. najprije se uočavaju relevantne fizikalne veličine [atributi], svojstva koja opisuju objekt i koja se mogu kvantificirati – izmjeriti u eksperimentu i izraziti brojkama
2. promjene tih fizikalnih veličina posljedica su djelovanja fizikalnih zakona [metode] – u fizici su to matematički modeli koji zadovoljavajuće opisuju ishode eksperimenata

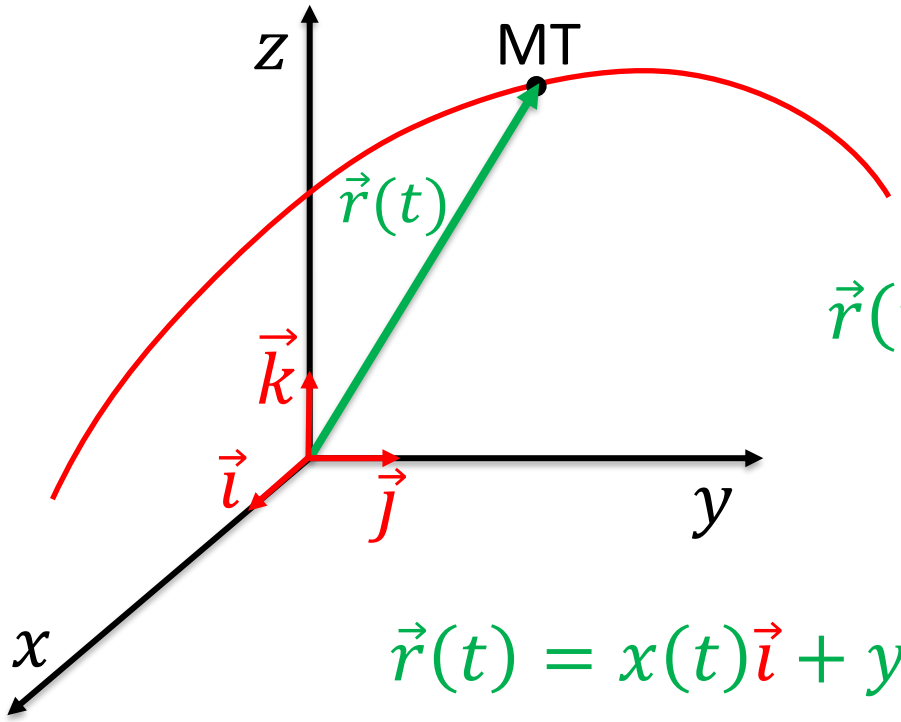
Primjer: mehanika materijalne točke

- radi jednostavnosti zadržat ćemo se na mehanici materijalne točke
- materijalna točka je definirana (potpuno određena) trima fizikalnim veličinama: položajem, brzinom i masom – to će ujedno biti atributi klase **`mt`** koja će opisivati materijalnu točku
- međuigra tih veličina određena je 2. Newtonovim zakonom

Položaj materijalne točke

- da bi mogli opisati gibanje materijalne točke moramo u svakom trenutku znati gdje se ona nalazi
- u 1D sustavu (tj. gibanju duž pravca) dovoljna je jedna koordinata koja određuje položaj (na primjer x)
- za opis 2D i 3D sustava najčešće se koristi takozvana r-metoda, tj. položaj materijalne točke određen je vektorom položaja $\vec{r}$
- vektoru položaja početak je u ishodištu koordinatnog sustava, vrh na trenutnoj poziciji materijalne točke

Vektor položaja u trenutku t



$\vec{r}(t)$ – vektor položaja
(radijus vektor)

$$\vec{r}(t) = x(t)\vec{i} + y(t)\vec{j} + z(t)\vec{k}$$

matematički, to je parametarska jednadžba krivulje (putanje)

Brzina materijalne točke

- zbog tromosti (inercije) i 1. Newtonovog zakona (zapravo Galilejevog zakona inercije) materijalna točka ima svojstvo da se giba stalnom brzinom $\vec{v}$ koja će se promijeniti tek ukoliko na materijalnu točku djeluje rezultantna sila
- brzina je mjera promjene položaja u vremenu, tj. ukoliko se u vremenskom intervalu Δt položaj promijeni za $\Delta\vec{r}$ srednja brzina definira se kao

$$\vec{v}_s = \frac{\Delta\vec{r}}{\Delta t}$$

Trenutna brzina

- trenutna brzina definira se kao limes kad vremenski interval teži ka nuli, tj. postaje beskonačno kratak

$$\vec{v} = \lim_{\Delta t \to 0} \frac{\Delta \vec{r}}{\Delta t}$$

- naša simulacija na računalu ne može imati beskonačno mnogo koraka pa ćemo se nadati da će rezultat biti barem približno točan ukoliko odaberemo da je vremenski interval Δt dovoljno kratak

Ubrzanje materijalne točke

- ubrzanje je mjera promjene brzine u vremenu, tj. ukoliko se u vremenskom intervalu Δt brzina promijeni za $\Delta\vec{v}$ srednje ubrzanje definira se kao

$$\vec{a}_s = \frac{\Delta\vec{v}}{\Delta t}$$

- trenutno ubrzanje je limes kad vremenski interval teži ka nuli, tj. postaje beskonačno kratak

$$\vec{a} = \lim_{\Delta t \to 0} \frac{\Delta\vec{v}}{\Delta t}$$

2. Newtonov zakon

- 2. Newtonov zakon je jednadžba gibanja, tj. opisuje da do ubrzanja, tj. promjene brzine, dolazi zbog djelovanja rezultantne sile, a promjeni brzine se opire svojstvo tijela koje se zove masa i mjera je tromosti tijela
- do ubrzanja $\vec{a}$ dolazi uslijed djelovanja sile $\vec{F}$:

$$\vec{a} = \frac{\vec{F}}{m} \quad \Leftrightarrow \quad \vec{F} = m\vec{a}$$

Eulerova metoda (1)

- ako je poznata brzina v materijalne točke mase m u trenutku t (pišemo $v(t)$), te sila F koja na nju djeluje, kolika je brzina u sljedećem trenutku $t + \Delta t$ (pišemo $v(t + \Delta t)$)?

$$a = \frac{\Delta v}{\Delta t} \Rightarrow \Delta v = a\Delta t \Rightarrow v(t + \Delta t) = v(t) + \Delta v$$

$$\Rightarrow v(t + \Delta t) = v(t) + a\Delta t = v(t) + \frac{F}{m}\Delta t$$

- za $\Delta t \to 0$ rezultat je egzaktan, za mali Δt približan

Eulerova metoda (2)

- slično i s položajem u sljedećem trenutku $t + \Delta t$

$$v = \frac{\Delta x}{\Delta t} \Rightarrow \Delta x = v\Delta t \Rightarrow x(t + \Delta t) = x(t) + v\Delta t$$

- dakle, ako iteriramo

```
this.v += dt * FR / this.m;
this.x += dt * this.v;
```

s dovoljno malim vremenskim koracima `dt`, trebali bi dobiti barem približan opis gibanja!

Implementacija

- najprije moramo definirati materijalnu točku: u konstrukturu zadajemo njezinu masu, te početni položaj i početnu brzinu (takozvane *početne uvjete*, engl. *initial conditions*)

```
let mtA = new MT(m, x0, v0);
```

- potom izračunamo rezultantnu silu FR koja djeluje na materijalnu točku i odaberemo vremenski korak

```
mtA.pomakni(dt, FR);
```

Klasa materijalna točka

```
class MT {
    constructor(m, x0, v0) {
        this.m = m;
        this.x = x0;
        this.v = v0;
    } // constructor

    // 2. Newtonov zakon
    pomakni(dt, FR) {
        this.v += dt * FR / this.m;
        this.x += dt * this.v;
    } // pomakni
} // class MT
```

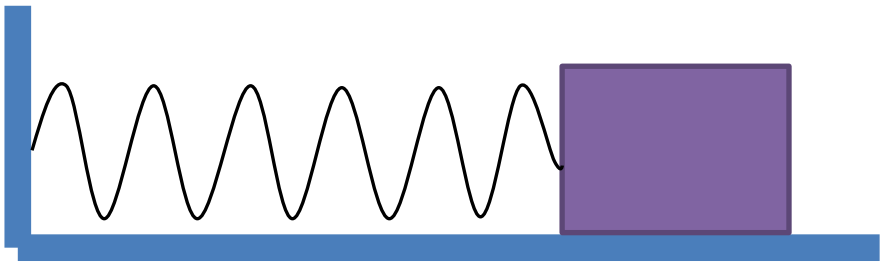
1D harmonijski oscilator

- povratna sila je linearno proporcionalna otklonu iz ravnotežnog položaja (Hookov zakon):

$$F = -kx$$

k je konstanta opruge

- ravnotežni položaj je $x = 0$



Kontrola: zakon očuvanja energije

- za konzervativne mehaničke sustave (sustave u kojima nema trenja) mora vrijediti zakon očuvanja mehaničke energije
- kinetička energija: $E_k = \frac{1}{2}mv^2$
- elastična potencijalna energija: $E_p = \frac{1}{2}kx^2$
- ukupna mehanička energija mora biti sačuvana:

$$E = E_k + E_p = \frac{1}{2}(mv^2 + kx^2) = konst.$$

Simulacija fizike u 3D (C++)

- položaj, brzina i sila su vektorske veličine
- ako implementiramo odgovarajuću klasu vektor3D i "overloadamo" operatore, rutine ostaju praktički iste

```
class mt {
  double m; // masa
  vektor3D r; // polozaj
  vektor3D v; // brzina
...
  void pomakni(double dt, vektor3D F) {
    v += F * (dt / m);
    r += v * dt;
  } // pomakni
```