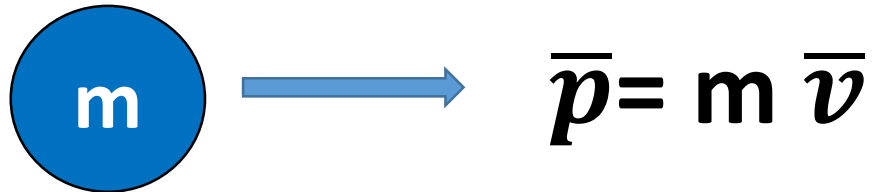


# Fysiikka

## viikko 4

- **Fysiikka:** Pallojen 3D törmäyksen mallinnus
- **Python:**
  - A) Interaktiivisuus hiiren avulla
  - B) taulukkomuuttujat Pythonissa
  - C) for silmukkarakenne Pythonissa
- **Tuntiesimerkit:**
  - 1) Drag and drop toiminto hiirellä
  - 2) Viisi palloa kuution sisällä
- **Tehtävä:** Biljardipelin koodin täydennys

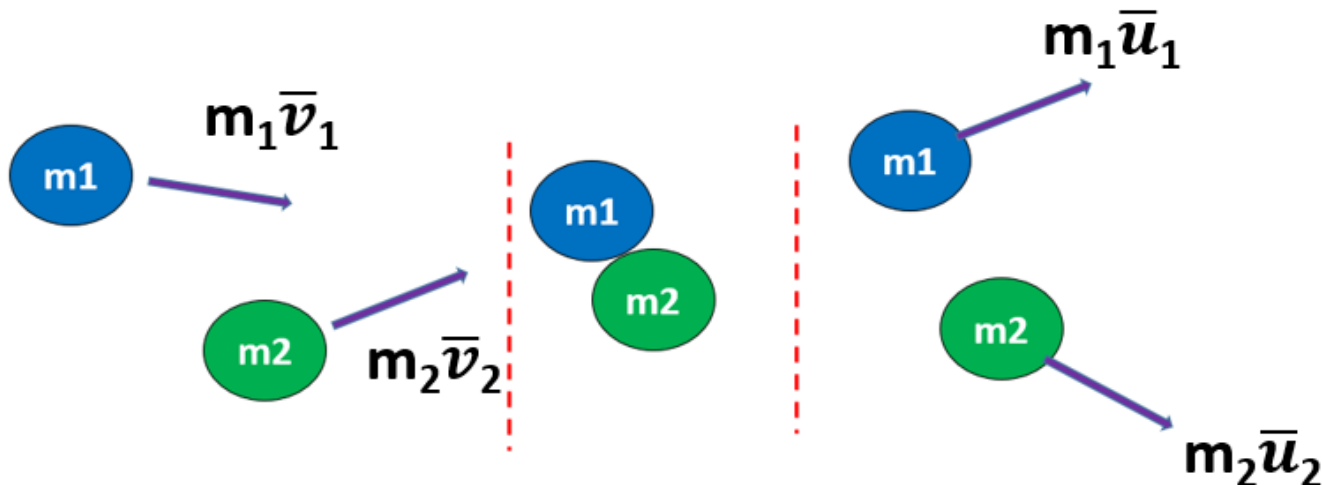
# Liikemäärän säilymislaki määrittää, mitä törmäyksissä tapahtuu


$$\bar{p} = m \bar{v}$$

Kappaleen liikemäärä  $\bar{p}$  on vektori, joka saadaan kertomalla kappaleen nopeusvektori kappaleen massalla

**Törmäyksissä kappaleiden liikemäärien summa säilyy**

$$m_1 \bar{v}_1 + m_2 \bar{v}_2 = m_1 \bar{u}_1 + m_2 \bar{u}_2$$

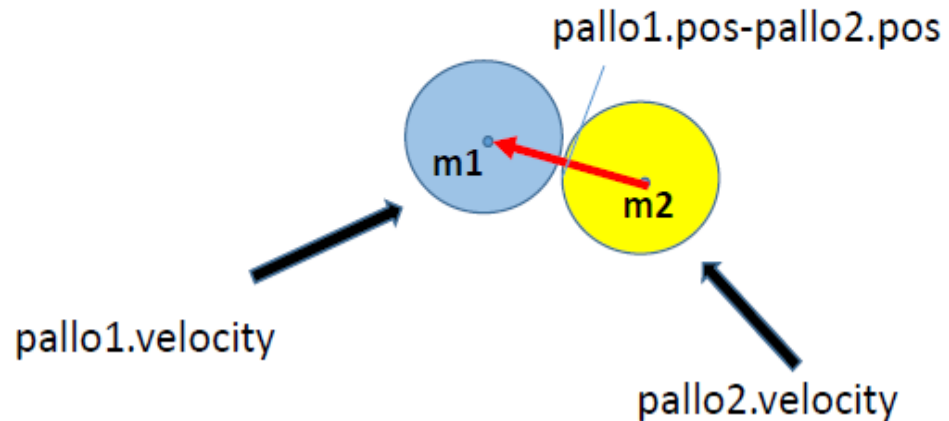


Seuraavissa kalvoissa esitetään tuloksia, jotka perustuvat tähän lakiin.

Tulosten johtaminen laista on suht. monimutkaista ja jätetään pois tästä esityksestä

# Kahden pallon 3D törmäys

## A. TÖRMÄYKSEN HAVAITSEMINEN



TÖRMÄYS TAPAHTUU, KUN PALLOJEN KESKIPISTEIDEN VÄLIVEKTORIN PITUUS  $<$  PALLOJEN SÄTEIDEN SUMMA

Lisäehtona, joka estää palloja tarttumasta toisiinsa, on että kimpoaminen tapahtuu vain kun pallot lähestyvät toisiaan.

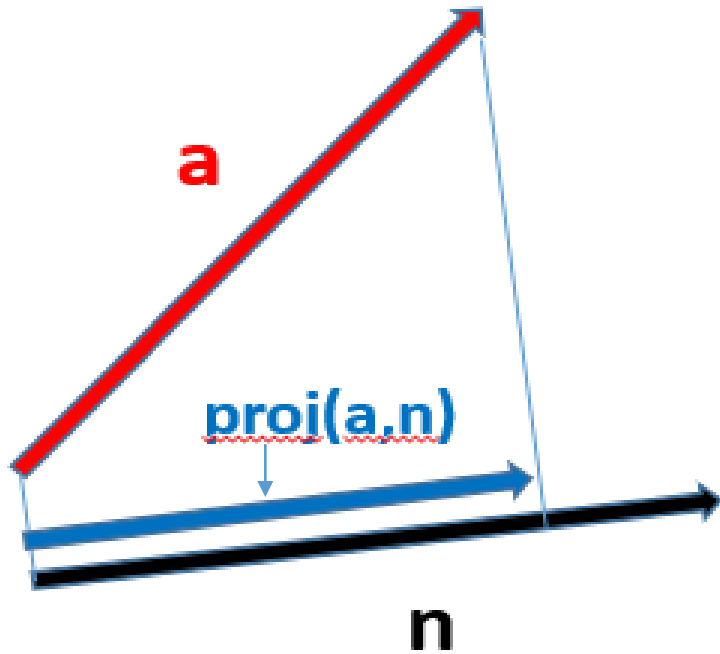
## B. MITÄ TÖRMÄYKSESSÄ TAPAHTUU

Oletetaan, että molemmat pallot liikkuvat.

Kimmoisassa törmäyksessä pallojen suhteellisen liikemäärän  $(=m_1\mathbf{v}_1 - m_2\mathbf{v}_2)$  kosketuspinnan normaalin  $\mathbf{n}$  suuntainen komponentti kääntyy päinvastaiseksi mutta säilyttää suuruutensa

Osittain kimmisessa törmäyksessä suunta kääntyy, mutta komponentin suuruus lyhenee kertoimella  $e$ .

# Tarvittavia vektoriluokan funktioita



**a:n** projektio **n:n** suuntaan : **proj(a,n)**

Vektorin **a** vektoriprojektio  
vektorin **n** suuntaan :  
**proj(a,n)**

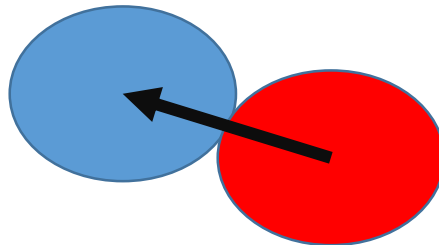
Vektorin **a** skalaariprojektio  
vektorin **n** suuntaan :  
**comp(a,n)**  
*= sama kuin mag(proj(a,n))*

Vektoreiden **a** ja **b** pistetulo  
**dot(a,n)** (ei tarvita tehtävässä)

# Törmäyksen havaitseminen

**Törmäysehto:** pallojen keskipisteiden välimatka  $<$  pallojen säteiden summa

```
if mag(pallo1.pos-pallo2.pos) < pallo1.radius+pallo2.radius:  
    törmäyskoodi ...
```



# Mitä tapahtuu. Merkitään kosketushetkellä kosketuspinnan normaalivektoria $n$ :llä

1. Lasketaan pallojen keskipisteiden suhteellinen nopeus

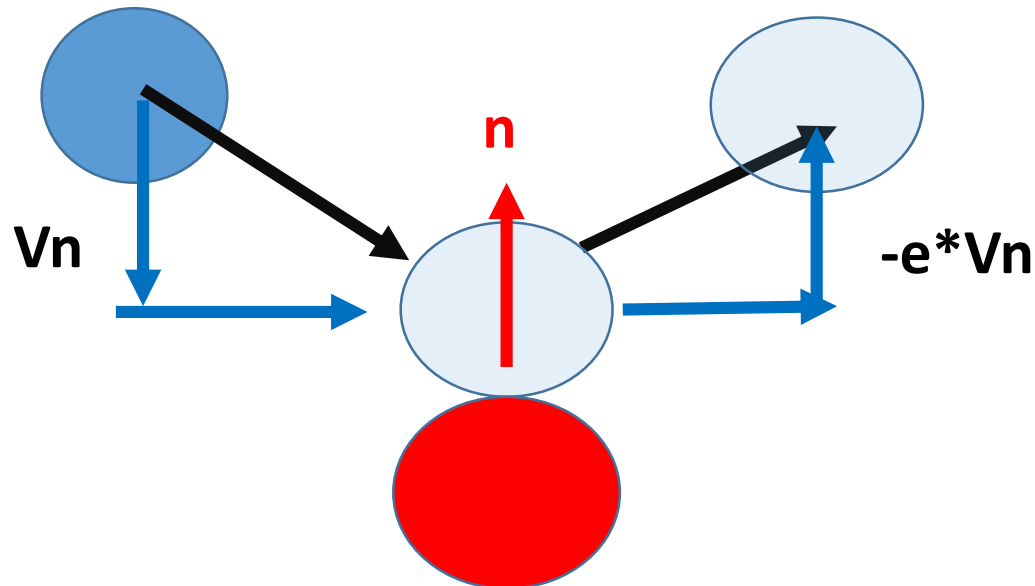
$$v_{rel} = \text{pallo1.velocity} - \text{pallo2.velocity}$$

2. Suhteellisen nopeuden  $v_{rel}$  kosketuspinnan suuntainen komponentti säilyy

3. Suhteellisen nopeuden normaalikomponentti kääntää suuntaa.

Jos törmäys on täysin kimminen, komponentin suuruus säilyy

Jos törmäys on osittain kimminen, suuruus kerrotaan kertoimella  $e$



Kerroin  $0 \leq e \leq 1$  kuvaa törmäyksen elastisuuden astetta.

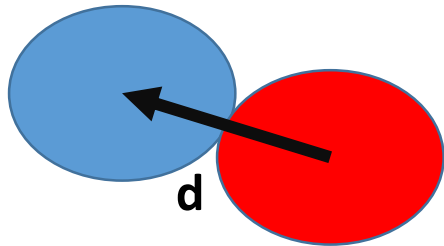
1 = täysin elastinen

0 = täysin kimmoton

Virallinen nimi englanniksi =  
"coefficient of restitution"

Törmäys toteutetaan animaatioissa lisäämällä  
while silmukan loppuun seuraava koodi

```
d=pallo1.pos-pallo2.pos                #lasketaan pallojen valivektori
vrel=pallo1.velocity-pallo2.velocity    #lasketaan nopeusero
If mag(d)< pallo1.radius+pallo2.radius and comp(vrel,d)<0:  #tormayksen havaitseminen
    pallo1.velocity+=- (1+e)*m2*proj(vrel,d)/(m1+m2)        #pallon1 uusi nopeus
    pallo2.velocity+= (1+e)*m1*proj(vrel,d)/(m1+m2)        #pallon2 uusi nopeus
```



Mikäli törmäyksessä on pinnan suuntaista kitkaa, tapahtuu asioita, joita ei todennäköisesti tavallisten peliohjelmistojen fysiikkamoottoreissa oteta huomioon.

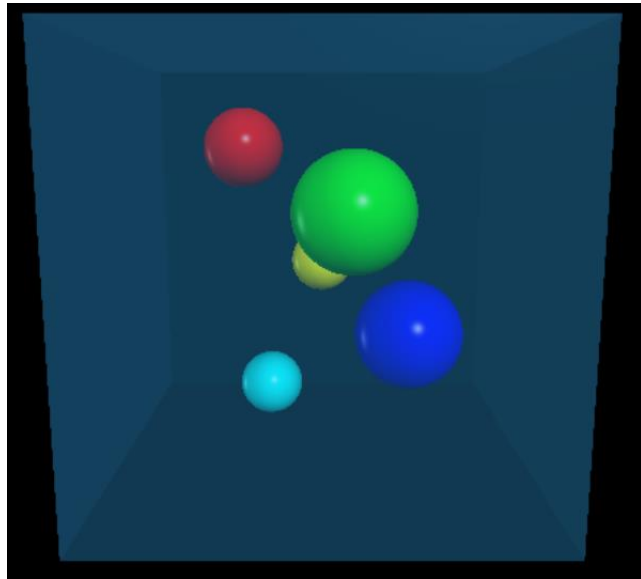
- Suhteellisen nopeuden kosketuspinnan suuntainen komponentti ei säily, vaan pienenee kertoimella  $1 - \mu e$ , missä  $\mu$  on pallojen välinen liukukitkakerroin
- Kitkan impulssi antaa molemmilla palloille pyörimisliikkeit, jonka kulmanopeudet on mahdollista laskea.

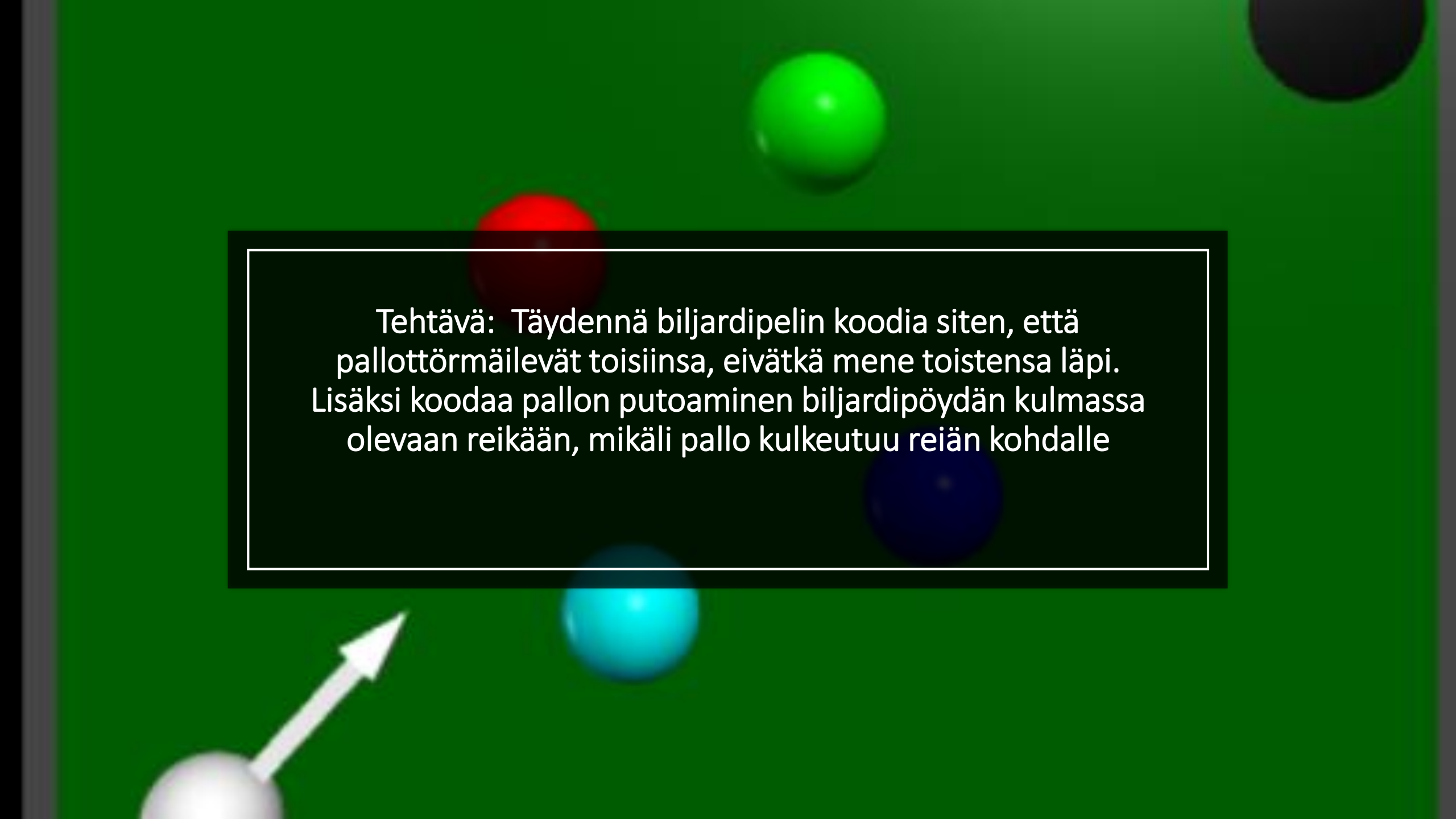




# Tuntiesimerkki: 5 eri kokoista, massaista ja väristä palloa törmäilee seiniin ja toisiinsa kuution sisällä

Lataa tuntiesimerkin koodin alkuosa Moodlesta kohdasta Teema3  
Tehdään valmiiksi tunnilla





Tehtävä: Täydennä biljardipelin koodia siten, että pallottörmäilevät toisiinsa, eivätkä mene toistensa läpi. Lisäksi koodaa pallon putoaminen biljardipöydän kulmassa olevaan reikään, mikäli pallo kulkeutuu reiän kohdalle

# Funktion määrittely Pythonissa

```
def funktionimi(argumentit):  
    lauseet  
    return paluuarvo (optional)
```

```
def f(x):  
    a=2  
    b=3  
    c = -5  
    return a*x**2+b*x + c
```

Määrittelee matemaattisen funktion  $f(x) = 2x^2 + 3x - 5$

testiohjelma

```
def f(x):  
    return 2*x**2 + 3*x-5  
print(f(2))
```

ajo

```
>>>  
9  
>>>
```

```
def change_g:  
    global g  
    g=1.6
```

Muuttaa pääohjelmassa määriteltyä g:n arvoa (uusi arvo 10)

testiohjelma

```
def change_g():  
    global g  
    g=1.6  
  
g=9.8  
print("vanha g:" ,g)  
change_g()  
print("uusi g:",g)
```

ajo

```
>>>  
( 'vanha g:', 9.8)  
( 'uusi g:', 1.6)  
>>> |
```

# Taulukkomuuttujat Pythonissa

```
luvut= [ 2 , 5, 3, 6, 7 ]
```

Luodaan taulukkomuuttuja, joka sisältää viisi lukua

```
luvut[0]
```

Taulukon luvut indeksoidaan alkaen 0:sta  
luvut[0] viittaa lukuun 2 (taulukon 1. lukuun)

Dynaamisen taulukon (linkitetty lista) teko on helppoa. Luodaan ensin tyhjä taulukko ja lisätään sitten siihen alkioita yksi kerrallaan

```
pallot= [ ]
```

komento luo tyhjän taulukon

Taulukkoon voi lisätä alkioita append metodilla

```
pallot.append= sphere(pos=vec(0,0,0),radius=2, color=color.red)
```

Lisää pallon taulukkoon

# range(n) ja for – silmukka Pythonissa

## range(n)

range(5)

## range(k, n)

range(2, 5)

- Tarkoittaa lukutaulukkoa [0,1,2,..., n-1]
- Tarkoittaa taulukkoa [0,1,2,3,4]
- Tarkoittaa lukutaulukkoa [k,..., n-1]
- Tarkoittaa taulukkoa [2,3,4]

## FOR SILMUKKA PYTHONISSA

```
for i in range(5):  
    print(i**2)
```

- For silmukassa, jossa indeksi saa peräkkäisiä arvoja, käytetään range funktiota.
- Esimerkki tulostaa lukujen 0,1,2,3 ja 4 neliöt:
  - 0,1,4,9,16

# Objektin siirtäminen hiirellä näytöllä paikasta toiseen vaatii funktioita drag, move ja drop

```
pallo=sphere(pos=vec(-23,-18,0),radius=3,color=color.white)

drag_pos=None
def grab(evt):
    global drag_pos
    if mag(pallo.pos-evt.pos)<6:
        drag_pos=evt.pos
        scene.bind('mousemove',move)
        scene.bind('mouseup',drop)

def move(evt):
    global drag_pos
    uusipos=evt.pos
    if uusipos!=drag_pos:
        siirtyma=uusipos-drag_pos
        drag_pos=uusipos
        pallo.pos+=siirtyma

def drop(evt):
    scene.unbind('mousemove',move)
    scene.unbind('mouseup', drop)

scene.bind('mousedown',grab)    #kutsutaan hiirella tarttumisfunktiota
```

**grab** – funktiossa kuvataan, miten hiirellä tartutaan objektiin

**move** – funktiossa kuvataan, miten hiirellä liikutetaan objektia

**drop** – funktiossa kuvataan, miten hiiren painikkeen vapauttaminen lopettaa objektin liikkumisen

**Funktioissa esiintyvät hiiren event :it ovat**

<b>mousedown</b>	painike alas
<b>mousemove</b>	hiiren liike
<b>mouseup</b>	painikkeen vapautus