Brick Breaker -peli Pythonilla

Jatketaan siitä mihin vaiheessa 1 jäätiin: ikkuna kasvaa, maila liikkuu, ja pallo on paikallaan. Tässä vaiheessa lisätään tiilet ja saadaan pallo liikkumaan ja kimpoamaan.

Kaikki uusi koodi lisätään ennen while-silmukkaa, paitsi pallon liike ja törmäystarkistukset, jotka lisätään silmukan sisälle. Muista tallentaa usein ja kokeile peliä aina askeleen jälkeen.

Vaihe 2: Tiilet ja pallon liike

Askel 1: Random-kirjasto ja piirtonopeus

Tuo random-kirjasto tiedoston alkuun turtle-kirjaston jälkeen. Sitä tarvitaan kohta pallon nopeuden arpomiseen. Muuta myös ikkunan tracer-arvo 0:sta 2:een: kun ruudulla on kohta paljon tiiliä, tämä nopeuttaa piirtämistä ja pitää pelin sulavana.

```
import random
```

Lisää seuraava rivi ikkuna.setup -rivin jälkeen:

```
ikkuna.tracer(2)
```

Askel 2: Tiilet listojen avulla

Tiiliä tulee ruudulle yhteensä 69 kappaletta (kolme riviä, 23 tiiltä per rivi), joten niitä ei kannata luoda yksitellen käsin. Sen sijaan käytetään listoja (taulukoita): jokaiselle riville oma lista, ja nämä kolme listaa kootaan vielä yhteen isompaan listaan nimeltä tiilikasa.

Lisää tämä ikkunan asetusten jälkeen, ennen px- ja py-muuttujia:

```
tiilet1 = []
tiilet2 = []
tiilet3 = []

tiilikasa = [tiilet1, tiilet2, tiilet3]

for tiilet in tiilikasa:
    for i in range(23):
        tiilet.append(turtle.Turtle())
```

Ensimmäinen for-silmukka käy läpi jokaisen kolmesta rivistä. Sisempi for-silmukka luo 23 uutta turtle-oliota ja lisää ne kyseiseen riviin append()-komennolla.

Tiilet ovat nyt olemassa, mutta niillä ei ole vielä väriä, muotoa eikä sijaintia. Lisää seuraavaksi asettelu:

```
posx = -28
posy = 150

for tiilet in tiilikasa:
    posx = -28
    for tiili in tiilet:
        tiili.speed(0)
        tiili.color("red")
        tiili.penup()
        tiili.shape("square")
        tiili.setpos(posx * 10, posy)
        posx += 2.5
    posy += 50
```

Sijainnin x-koordinaatti posx kasvaa jokaisen tiilen jälkeen, jolloin tiilet asettuvat vierekkäin, ja posy kasvaa aina kun siirrytään seuraavalle riville, jolloin rivit pinoutuvat päällekkäin.

Askel 3: Pallolle nopeus

Etsi pallon alkukoordinaatit (x = 0 ja y = -200) ja lisää niiden perään kaksi uutta riviä, ennen pallo-olion luontia:

```
vx = random.randint(3, 8)
vy = random.randint(5, 8)
```

Muuttujat vx ja vy kertovat, kuinka monta pikseliä pallo liikkuu joka kierroksella x- ja y-suunnassa. random.randint(3, 8) arpoo kokonaisluvun väliltä 3-8, joten jokaisella kerralla pallo lähtee hieman eri nopeudella liikkeelle.

Askel 4: Pallon liike ja törmäykset pääsilmukkaan

Päivitä pääsilmukka seuraavanlaiseksi.

```
while True:
    leveys += dx
    korkeus += dy
    if leveys > 600 and korkeus > 600:
        dx = 0
        dy = 0

    if pallo.xcor() > 280 or pallo.xcor() < -280:
        vx = -vx
    if pallo.ycor() > 280 or pallo.ycor() < -280:
        vy = -vy

    if (pallo.xcor() < maila.xcor() + 70
        and pallo.xcor() > maila.xcor() - 70
        and pallo.ycor() <= maila.ycor() + 10):
        vy = -vy

    x = pallo.xcor()
    x += vx
    y = pallo.ycor()
    y += vy
    pallo.setpos(int(x), int(y))

    ikkuna.setup(leveys, korkeus)
    ikkuna.update()
```

Ensimmäiset kaksi if-lausetta kääntävät pallon suunnan, kun se osuu ikkunan reunaan. Kolmas if-lause tarkistaa, osuuko pallo mailaan: jos pallon x-koordinaatti on tarpeeksi lähellä mailan x-koordinaattia JA pallo on tarpeeksi alhaalla, pallo kimpoaa ylöspäin. Lopuksi pallon uusi sijainti lasketaan vx:n ja vy:n avulla ja asetetaan setpos()-komennolla.

Kokeile: Pallon pitäisi nyt liikkua, kimmota ikkunan reunoista, ja kimmota myös mailasta kun osut siihen. Tiilet eivät vielä reagoi palloon.

Kustomoi oma pelisi: Tiiliä ja palloa voi muokata samalla tavalla kuin vaiheessa 1:
- Tiilien väri: vaihda tiili.color("red") esim. muotoon tiili.color("orange") tai anna jokaiselle riville oma väri
- Pallon nopeus: muuta random.randint(3, 8) -lukuja suuremmiksi tai pienemmiksi, jos haluat hitaamman tai nopeamman pallon