13. NODARBĪBA. PYTHON TURTLE

ANIMĀCIJA AR PYTHON

Animācijas pamatā ir secīgi attēli, kas tiek rādīti tik ātri, ka rodas nepārtrauktas kustības iespaids. Vienu attēlu sauc par kadru. Katrs kadrs ir nedaudz atšķirīgs no iepriekšējā. Kadri jārāda ar ātrumu apmēram 12 kadri sekundē. Mūsdienās filmās izmanto 24 kadrus sekundē.

Aplūkosim, ka veidot vienkāršu animāciju ar *Python* moduli **turtle**. Animācijā pārvietosim kvadrātu pāri ekrānam no kreisās uz labo pusi.

Kvadrāta animācija

Lai radītu ilūziju, ka kvadrāts pārvietojas pa ekrānu no kreisās uz labo pusi, nepieciešams uzzīmēt ļoti daudz kvadrātu – katrs nākamais kvadrāts tiek zīmēts nedaudz pa labi no iepriekšējā, un šie kvadrāti jārāda ļoti ātri.

Animācijas radīšanas algoritms – izdzēst iepriekšējo kvadrāta zīmējumu, pēc tam uzzīmēt jaunu kvadrātu nedaudz pa labi no iepriekšējā.

Programmu sāksim ar to, ka uzzīmēsim kvadrātu. Lai kods būtu labāk lasāms un saprotamāks, definēsim funkciju kvadrāta uzzīmēšanai.

Iestatīsim noteiktu ekrāna platumu, lai varētu pārvietot kvadrātu pāri ekrānam no kreisās uz labo pusi.

Pēc tam realizēsim iepriekš aprakstīto algoritmu, lai radītu kvadrāta kustības ilūziju.

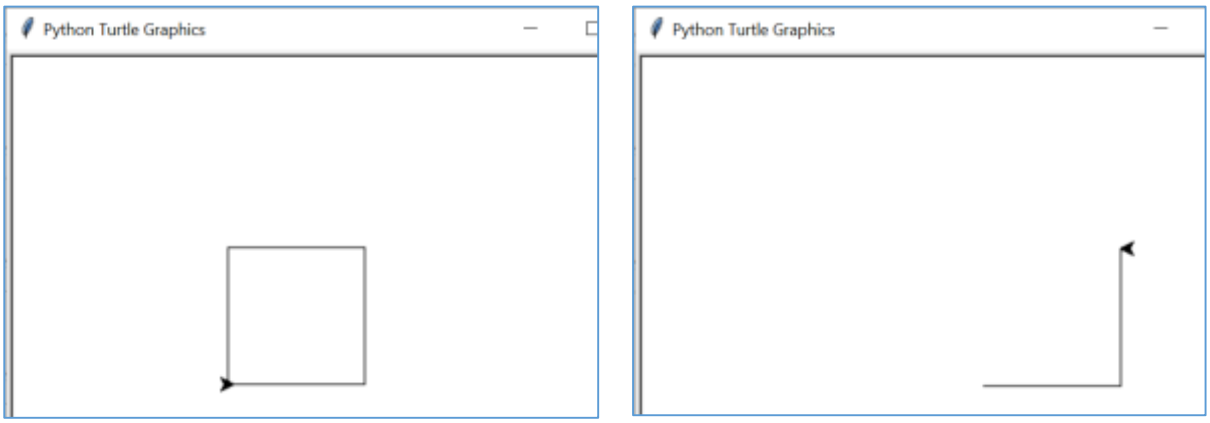
Programmas kods:

```
import turtle

def draw_square(d) :              # funkcija uzzimee kvadratu
   for side in range(4) :
      d.forward(100)
      d.left(90)

screen = turtle.Screen()          # izveido jaunu zimejuma logu
screen.setup(500,500)             # 500 x 500 loga izmeri

felix = turtle.Turtle()
```

```
felix.speed(0)                  # iestata atrumu

felix.penup()
felix.goto(-350, 0)             # ziimeesanu saak aarpus ekrana
felix.pendown()

k = 4                           # kvadraata parvietojums starp kadriem
for i in range(int(600 // k)):
    felix.clear()               # nodzes ieprieksejo ziimejumu
    draw_square(felix)          # zime kvadratu
    felix.forward(k)            # parvieto nedaudz uz prieksu

screen.exitonclick()
```

Ekrāna atjaunināšana

Palaižot programmu, mēs varam redzēt, ka kvadrāts pārvietojas pāri ekrānam, bet animācija izskatās saraustīta. *Python* ekrāns (*screen*) automātiski nosaka, kad atjaunināt attēlu, un parasti tas notiek kvadrāta zīmēšanai pa vidu. Tādejādi katrā kadrā redzama daļa no kvadrāta.



Lai novērstu raustīšanos, programmai jāseko, lai kadrs tiktu parādīts tikai tad, kad tas ir uzzīmēts. Šim nolūkam vispirms jāpasaka ekrānam, lai neko nerāda automātiski. To var izdarīt ar **screen** metodes **tracer** palīdzību:

```
screen.tracer(0)
```

Ja funkcijas parametrs n ir 0, automātiskā ekrāna atjaunināšana ir izslēgta (ja n ir 1, automātiskā ekrāna atjaunināšana ir ieslēgta). Ja ekrāna automātiskā atjaunināšana ir izslēgta, tad, lai tiktu atspoguļots pašreizējais zīmējums, tieša veidā ir jāizsauc **screen** metode **update**, kas veiks ekrāna atjaunināšanu.

Ja vēlamies parādīt kvadrātu katru reizi, kad ir pabeigta tā zīmēšana, pēc funkcijas, kas uzzīmē kvadrātu, izsaukšanas, ir jāizsauc ekrāna metode **update**:

```
screen.update()
```

Ātruma kontrole

Vēl programmā jāveic viena izmaiņa – nākamais kvadrāts jāpārvieto par daudz mazāku attālumu.

*Tas ir tāpēc, ka pirmajā kvadrāta zīmēšanas piemērā ekrāns (**screen** objekts) kadrus parādīja automātiski – tas izmantoja daudzus kadrus, lai parādītu katra kvadrāta uzzīmēšanu. Kadru skaits vienam kvadrātam varētu būt apmēram 10. Līdz ar to kvadrāta pārvietošana notika apmēram katrā desmitajā kadrā. Ja animācijas kadru ātrums ir apmēram 10 kadri sekundē, tas nozīmē, ka kvadrāta pārvietošana pa labi notiek apmēram vienu reizi sekundē.*

Ja mēs izvēlamies parādīt jaunu kadru tikai pēc tam, kad pabeigta kvadrāta zīmēšana, tad ātrums 10 kadri sekundē nozīmē 10 kvadrāta pārvietojumus. Līdz ar to izskatīsies, ka kvadrāts pārvietojas ļoti ātri. Tāpēc, lai palēninātu kvadrāta pārvietošanu, ir jāsamazina attālums, par kādu nākamais kvadrāts tiek pārvietots pret iepriekšējo.

Visbeidzot paslēpsim pašu bruņrupuci, lai varam redzēt tikai kvadrāta zīmējumu. Un palielināsim kvadrāta zīmējuma līniju biezumu.

Programmas kods:

```
import turtle

def draw_square(d) :       # funkcija uzzimee kvadratu
   for side in range(4) :
      d.forward(100)
      d.left(90)

screen = turtle.Screen()
screen.setup(500,500)
screen.tracer(0)          # neatjaunot attelu automaatiski

felix = turtle.Turtle()
felix.speed(0)
felix.width(3)
felix.hideturtle()        # paslepj rupuci, lai redzetu tikai zimejumu

felix.penup()
felix.goto(-350, 0)
felix.pendown()
```

```
k = 0.04                          # kvadraata parvietojums starp kadriem
for i in range(int(600 // k)):
    felix.clear()                 # izdzes iepriekšejo zimejumu
    draw_square(felix)
    screen.update()               # parada jau uzzimetu kvadratu (viens kadrs)
    felix.forward(k)

screen.exitonclick()
```

Veidojot animāciju var izmantot .gif attēlu Citi attēla formāti netiek pieļauti.

Programmas koda piemērs

```
import turtle

screen = turtle.Screen()
screen.setup(500,500)
screen.tracer(0)
screen.addshape("butterfly.gif")    # registree attelu, lai to var izmnatot kaa formu

felix = turtle.Turtle()
felix.speed(0)
felix.shape("butterfly.gif")        # pieskir rupucim formu (attelu)

felix.penup()
felix.goto(-350, 0)

k = 0.04
for i in range(int(700 // k)):
    screen.update()
    felix.forward(k)

screen.exitonclick()
```