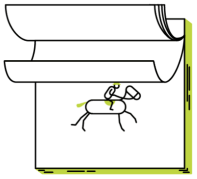
4. Animationen

Lernziele:

- Du lernst, wie man Animationen programmiert und wie man gewisse Grafikbestandteile im Bildbuffer speichern und löschen kann.
- Du wendest gelernte Konzepte wie Schleifen, Befehle und Parameter in Projekten an.



Das Prinzip der bewegten Bilder kennen wir wohl alle in Form des bekannten „Daumenkinos“, bestehend aus kleinen Bildern, deren Position und/oder Form sich auf jedem Blatt ein wenig ändert. Schnelles „Durchblättern“ erzeugt die Illusion einer tatsächlichen Bewegung.

Dieses Prinzip übertragen wir nun auf unsere Programme:

4.1 Animierte Grafiken programmieren

Für eine Animation wiederholen wir die folgenden Schritte beliebig oft:

1. Ein Bild zeichnen.
2. Kurz warten, damit man es erkennen kann.
3. Das Bild löschen (Alternative: mit der Hintergrundfarbe übermalen).
4. Die Position/Ausrichtung des Bildes ändern.

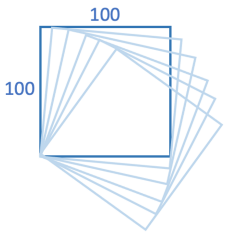
Damit die Schritte programmieren können, benötigen wir zwei neue Befehle:

`delay(t)`	`t` Millisekunden warten bis zur Ausführung der nächsten Zeile
`clear()`	Löscht alle Grafiken und versteckt die Turtle (die an der aktuellen Position bleibt).
`clear(farbe)`	Setzt zusätzlich eine Hintergrundfarbe (z.B `clear("red")`).

Beispiel 4.1

Ein Quadrat mit Seitenlänge 100 soll sich um seine linke untere Ecke drehen.

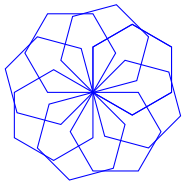
```
1  from gturtle import*
2
3  def quadrat():
4      repeat 4:
5          fd(100)
6          rt(90)
7
8  ################
9  makeTurtle()
10 ht()
11
12 repeat:
13     quadrat()
14     delay(50)
15     clear()
16     rt(6)
```



Mit `repeat` (ohne Zahl) wird der Schleifenkörper unendlich oft wiederholt. Auf Zeile 14 wartet das Programm 50 ms, bis es Zeile 15 wechselt und die Grafik löscht (Man könnte die Grafik auch weiss übermalen mit `setPenColor("white")` und `dot(110)`). Die Turtle wird auf Zeile 16 leicht gedreht und der Prozess beginnt wieder von vorne (Zeile 13).

Aufgabe 4.1

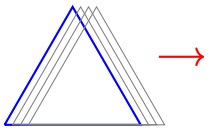
Schreibe ein Programm, mit dem die Figur, bestehend aus acht regelmässigen 6-Ecken, um den Mittelpunkt gedreht wird.



Aufgabe 4.2

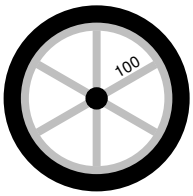
Schreibe ein Programm, das ein gleichseitiges Dreieck vom linken Bildschirmrand horizontal zum rechten Bildschirmrand bewegt.

Mit `setPos(-450,0)` kannst du die Startposition der Turtle horizontal um 450 Pixel nach links verschieben.



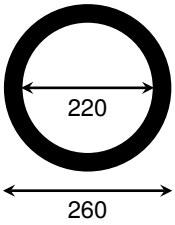
Aufgabe 4.3

Ein Programm soll ein drehendes Rad mit Reifen und Felgen zeichnen. Die gleichzeitige horizontale Bewegung folgt später im Kapitel Variablen.

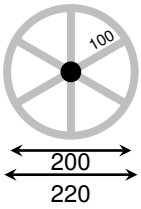


`rad()`

Zum Zeichnen zerlegen wir das Rad in seine Bestandteile mit den folgenden Massen:



`reifen()`



`felge()`

a) Definiere zuerst einen Befehl `reifen()`. Lege mit `dot(d)` eine weisse Kreisfläche über eine schwarze. Die Füllfarbe kannst du mit `setPenColor("black")` definieren.

b) Erstelle einen Befehl `felge()`, welche die graue runde Felge mit den Querstreben und dem kleinen schwarzen Kreis zeichnet. Die Breite der Felgen beträgt 10 und der Durchmesser des schwarzen Kreises 30.

c) Der Befehl `rad()` besteht nun aus den beiden Befehlen `reifen()` und `felge()`. Erstelle mit `rad()` die Animation so, dass sich das Rad in 3 Sekunden einmal dreht.

4.2 Hintergrund speichern

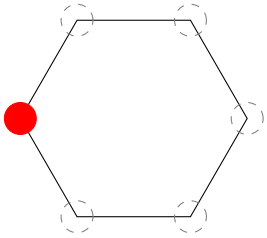
Bei aufwändigeren Animationen möchte man gewisse Objekte nicht bewegen (z.B. den Hintergrund). Dafür gibt es einen Befehl, der alles, was bis zu dessen Aufruf gezeichnet wird, in einen sogenannten **Bildbuffer** speichert. Die gespeicherten Grafiken werden beim Aufruf von `clear()` nicht gelöscht, bleiben also fixiert.

`savePlayground()`	Speichert alle vor dem Aufruf dieses Befehls programmierten Grafiken

Beispiel 4.2

In diesem Beispiel soll ein roter Kreis von Ecke zu Ecke eines regelmässigen Sechsecks „springen“ und das Sechseck an Ort und Stelle bleiben:

```
1  from gturtle import*
2
3  def sechseck():
4      repeat 6:
5          fd(100)
6          rt(60)
7
8  def Kreis():
9      setPenColor("red")
10     dot(30)
11 ########################
12 makeTurtle()
13 ht()
14
15 sechseck()
16 savePlayground()
17
18 repeat:
19     Kreis()
20     delay(1000)
21     clear()
22     pu()
23     fd(100)
24     rt(60)
25     pd()
```



Das gezeichnete Sechseck (Zeile 15) wird im Bildbuffer (Zeile 16) gespeichert.

Mit `clear()` (Zeile 21) wird der (auf Zeile 19) gezeichnete Kreis gelöscht.

Aufgabe 4.4

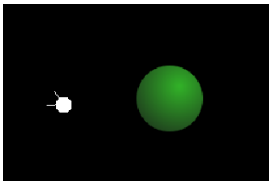
Erstelle ein Programm für eine Sekunden-Stoppuhr. Verwende das Zifferblatt aus Aufgabe 2.7b) und definiere einen Befehl `zifferblatt()` sowie einen Befehl `zeiger()` für einen roten Sekundenzeiger. (Der graue Rand ist optional)

Für die Animation kann das Zifferblatt im Hintergrund gespeichert werden.

Optional: Füge zusätzlich einen Minutenzeiger hinzu.

Aufgabe 4.5

a) Ein kleiner Satellit soll um einen Planeten (grün) kreisen. Erstelle je einen Befehl für beide Objekte.
Mit `drawImage("sprites/marble_1.png")` lässt sich der grüne Planet aus der Bilddatenbank einfügen.



b) Erweitere das Programm mit einem weiteren Satelliten, der sich in die gleiche Richtung um den Planeten dreht.