Programmieren mit jython – Ergänzung: Mausevents und mehr

Lernziele

- Ereignisgesteuerung über Maustaste

Bisher bestand ein Programm aus einem einzigen Ablaufstrang, in dem eine Anweisung um die andere mit möglichen Verzweigungen und Wiederholungen ausgeführt wird. Klickst du eine Maustaste, so weisst du aber nicht, wo sich dein Programm eben gerade befindet. Um das Klicken im Programm zu erfassen, muss daher ein **neues Programmierkonzept** benutzt werden, das **Ereignisgesteuerung** heisst.

Für das Erfassen von Mausevents stehen dir in makeTurtle() folgende Parameter zur Verfügung:

mousePressed	Maustaste wird gedrückt
mouseReleased	Maustaste wird losgelassen
mouseClicked	Maustaste wird gedrückt und losgelassen
mouseDragged	Maus wird mit gedrückter Taste bewegt
mouseMoved	Maus wird bewegt

Vorgehen: Du definierst eine Funktion mit irgendeinem Namen, z.B. *hitMe()*, die im Programm nirgends explizit aufgerufen wird. Nun verlangst du vom Computer, dass **er** diese Funktion aufrufen soll, wenn die Maustaste geklickt wird. Du sagst also im Programm: *Wann immer die Maus geklickt wird, führe hitMe() aus*.

```
from gturtle import *
def hitMe(x, y):
    fill(x, y)

makeTurtle(mouseHit = hitMe)
hideTurtle()

repeat 12:
    repeat 6:
        forward(80)
        right(60)
    left(30)
```

1) Schreibe nebenstehendes Programm ab. Klicke in ein beliebiges Areal und beobachte was passiert.

2) Schreibe ein Programm, welches zufällig grosse Punkte nach einem Mausklick zeichnet. Damit die Mausposition interpretiert werden kann, muss die Funktion, welche nach dem Mausklick ausgeführt werden soll, u.a. der Befehl ‚setPos(x,y)' enthalten sein. Zuerst soll das Programm aber Fragen, ob die Punkte rot, grün, gelb oder schwarz gezeichnet werden sollen. Verwende dazu die Befehle if, elif und else.

3a) Verwende in einer Funktion ‚klick' den Befehl moveTo(x,y) und rufe dieselbe Funktion mit einem Mausklick auf.
3b) Erweitere das Programm derart, dass zudem bei jedem Mausklick zunehmend grössere Punkte zeichnen. Wenn die Punkte zu gross werden, soll wiederum bei einem Startwert der Punktgrösse von 10 begonnen werden. Verwende u.a. den Begriff global. Schau dazu in der Python-Referenz nach.

Für die Schnellen Aufgaben 4+5:
4) Verwende das Event ‚mouseMoved' sowie den Befehl moveTo um ein Programm zu schreiben, welches dem Mauszeiger folgt. Hinweis: um z.B. die x-Koordinate in gturtle-Koordinaten umzurechnen, wird folgendes Konstrukt benötigt: x = toTurtleX(event.getX())

5) Schreibe ein Programm, welches Dich zuerst nach der Anzahl der zu zeichnenden Punkte fragt und diese dann zufällig auf der Zeichenebene einzeichnet. Ein Drittel der Punkte soll rot, ein Drittel grün und der restliche Drittel blau eingefärbt werden. Das Programm soll eine Rückmeldung geben(z.B. ‚fertig'), nachdem es alle Punkte gezeichnet hat.

6)
a)Ein Punkt soll zufällig auf der Zeichenebene erscheinen und dann nach einer Sekunde wieder verschwinden (verwende die Befehle Turtle.sleep(1000) sowie ‚clear'). Dieser Vorgang soll zehnmal wiederholt werden.

b) Es soll probiert werden, den zufällig erscheinenden Punkt mit der Maus anzuklicken. Falls dies gelingt so erhält man einen Punkt, falls nicht sollen zwei Punkte vom Saldo abgezogen werden. Konzipiere das Programm derart, dass die Funktion punkte() zufällig einen Punkt zeichnet, eine andere Funktion soll auf Mausklicks reagieren (‚klick(x,y)') . Im repeat-loop soll dann getestet werden, ob der Mausklick auf dem zufälligen Punkt lag (d.h. also, dass die Position des Punktes sowie des Mausklicks bekannt sein müssten)

c) Erweitere das Programm derart, dass nach der Anzahl der Punkte, Zeitintervalle zwischen den Punkten etc. gefragt würde. Auch kann – je nachdem wie das Programm konzipiert wurde – der Punktestand überlistet werden, indem z.B. zweimal auf den gleichen Punkt geklickt wird.

7) Pfeilsteuerung
Die Schildkröte soll nun per Tastaturbefehle gesteuert werden. Übernimm folgende Zeilen und überlege Dir, was wo ausgeführt wird. Was bedeuten die Befehle ‚setHading' sowie ‚wrap()' ?

```
from gturtle import *

LEFT = 37
RIGHT = 39
UP = 38
DOWN = 40

def onKeyPressed(key):
    if key == LEFT:
        setHeading(-90)
    elif key == RIGHT:
        setHeading(90)
    elif key == UP:
        setHeading(0)
    elif key == DOWN:
        setHeading(180)

makeTurtle(keyPressed = onKeyPressed)
wrap()
while True:
     forward(10)
```

Ergänze das Programm derart, dass zum Start ein Labyrinth gezeichnet werden soll und die Turtle hindurch navigiert werden soll. Idealerweise kann die Turtle nicht durch die Labyrinthwände