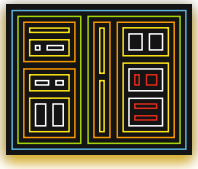
Beispiellösung: Rechteckschoner

Als eine ältere Mitschülerin von den letzten Zweitrundenaufgaben erzählte, hatte Moni die Idee zu einem eigenen Bildschirmschoner: Gezeichnet werden soll ein Rechteck, das zwei Rechtecke enthält, die jeweils wieder zwei Rechtecke enthalten usw. Damit das nett aussieht, soll bei der Rechteckschachtelung der Zufall eine sinnvolle Rolle spielen, und ein enthaltenes Rechteck soll sich nicht nur in der Größe von dem es unmittelbar enthaltenden Rechteck unterscheiden – etwa so:



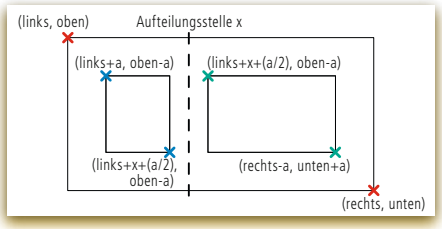
Nach dem Aufbau soll das fertige Rechteckschachtelbild immer wieder so aktualisiert werden, dass mal das eine, mal das andere Rechteck (evtl. mit allen enthaltenen Rechtecken) sein Aussehen verändert.

Aufgabe

Entwickle einen solchen „Rechteckschoner". Demonstriere seine Funktion an mindestens drei verschiedenen Rechteckschachtelbildern mit jeweils zwei Veränderungen.

Lösungsidee

Um ein einzelnes Rechteck zeichnen zu können, müssen seine Position und seine Farbe festgelegt werden. Die Farbe wird zufällig aus einer festen Liste gewählt. Die Position eines Rechtecks wird durch zwei diagonal gegenüberliegende Eckpunkte festgelegt. Beim äußersten Rechteck werden die Eckpunkte so bestimmt, dass sie alle den gleichen Abstand zu den Bildschirmrändern haben. Um zwei Rechtecke in das äußere zu zeichnen, wird dessen Fläche zufällig aufgeteilt. Die Eckpunkte der inneren Rechtecke werden so bestimmt, dass sie den gleichen Abstand zueinander haben wie zum äußeren Rechteck. Die folgende Abbildung zeigt, wie die Eckpunkte der inneren Rechtecke aus den Eckpunkten des äußeren Rechtecks (links, oben) und (rechts, unten) sowie der „Aufteilungsstelle" x berechnet werden. Das äußere Rechteck wird senkrecht aufgeteilt, wenn es breiter ist als hoch (wie in der Abbildung); andernfalls wird es waagerecht aufgeteilt, und die Eckpunkte der inneren Rechtecke werden entsprechend berechnet.



Damit das Rechteckbild gleichmäßig gefüllt wird, wird eine Liste der noch zu zeichnenden Rechtecke geführt. Diese Liste wird nach und nach abgearbeitet. Beim Füllen eines Rechtecks werden die beiden inneren Rechtecke hinten an die Liste angehängt. So wird erreicht, dass die Rechtecke Stufe für Stufe gezeichnet werden: erst das äußere Rechteck, dann dessen zwei innere Rechtecke, dann deren vier innere Rechtecke usw. Da ein Rechteck nur dann gefüllt wird, wenn es noch genügend Platz für zwei innere Rechtecke hat, endet dieser Prozess, wenn alle zu zeichnenden Rechtecke die Mindestgröße unterschreiten.

Alle gezeichneten Rechtecke werden in einer Liste abgespeichert. Ist das Bild einmal fertig, wird aus dieser Liste zufällig ein Rechteck ausgewählt. Es wird dann mit seinem Inhalt zunächst komplett gelöscht und übermalt und dann mit neuen inneren Rechtecken neu gezeichnet – auf die gleiche Art und Weise wie ursprünglich das äußerste Rechteck.

Umsetzung

Die Lösungsidee wird in der Sprache Python umgesetzt, und zwar mit Hilfe des Moduls „turtle", das viele Funktionen zum Zeichnen von Figuren bereitstellt.

Alle Eigenschaften und wichtigen Funktionen eines Rechtecks werden in einer Klasse **Rechteck** zusammengefasst. Eigenschaften sind im Wesentlichen die Koordinaten der Eckpunkte (**links**, **oben**, **rechts** und **unten**) und die Farbe **farbe**. Als Funktionen (oder besser: Methoden) eines Rechtecks werden definiert:

zeichne : nutzt Funktionen des Moduls turtle, um anhand der Eckpunkte ein Rechteck der gesetzten Farbe zu zeichnen.

fuelle : bestimmt, ob das Rechteck groß genug ist, um zwei innere Rechtecke zu erhalten. Wenn ja, wird das Rechteck aufgeteilt und es werden zwei innere Rechtecke erzeugt.

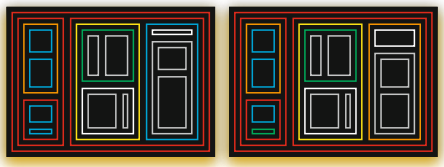
loesche : entfernt das Rechteck aus der Liste aller Rechtecke und löscht rekursiv seine Inhaltsrechtecke; jedes gelöschte Rechteck wird abschließend in Hintergrundfarbe neu gezeichnet und damit auch aus der Anzeige gelöscht.

Alle Rechtecke (also alle Objekte der Klasse **Rechteck**) werden in der globalen Liste **alle_rechtecke** gespeichert.

Die zentrale Funktion zum Zeichnen eines Rechtecks ist **zeichne_rechtecke**; sie benutzt die lokale Liste **aktuelle_rechtecke**, um – wie in der Lösungsidee beschrieben – die Rechtecke stufenweise zu erzeugen und zu zeichnen.

Beispiel

Aus Platzgründen zeigen wir nur ein Rechteckschachtelbild mit einer Veränderung. Die Bilder sind 400 mal 300 Pixel groß. Die Veränderungen vom linken zum rechten Bild sind: Das innerste Rechteck links unten wurde gelöscht und in anderer Farbe neu gezeichnet; außerdem wurde im rechten Drittel des Bildes das blaue Rechteck gelöscht und mit neuen inneren Rechtecken neu gezeichnet.



Quelltext

```
# Modulimport
import turtle
import random

# Hier kann die Größe des Fensters an den Bildschirm angepasst werden.
SCREEN_WIDTH = 400
SCREEN_HEIGHT = 300

SPEED = 5 # Zeichengeschwindigkeit

# Farben für die Rechtecke
COLORS = ["red", "white", "yellow", "green", "blue",
      "orange", "brown", "pink", "gray", "violet"]
BGCOLOR = "black" # Hintergrundfarbe

# Rechteck-Klasse mit Eigenschaften und Operationen
class Rechteck:

  def __init__(self,links,oben,rechts,unten):
    '''Die Eigenschaften des Rechtecks werden initialisiert.'''
    self.links=links
    self.oben=oben
    self.rechts=rechts
    self.unten=unten
    self.hoehe=oben-unten # nicht nötig, aber nützlich
    self.breite=rechts-links # nicht nötig, aber nützlich
    self.farbe=self.waehle_farbe()
    self.inhalt=[] # Liste für die beiden direkt enthaltenen Rechtecke

  def waehle_farbe(self):
    '''Die Linienfarbe wird zufällig ausgewählt.'''
    random.shuffle(COLORS)
    stift.color(COLORS[0])

  def zeichne(self):
    '''Das Rechteck wird mit Turtle-Kommandos gezeichnet.'''
    stift.up()
    stift.goto(self.links,self.oben)
    stift.down()
    stift.fd(self.breite)
    stift.right(90)
    stift.fd(self.hoehe)
    stift.right(90)
    stift.fd(self.breite)
    stift.right(90)
    stift.fd(self.hoehe)
    stift.right(90)

  def fuelle(self):
    '''Das Rechteck bekommt zwei Rechtecke als Inhalt, wenn es groß
    genug ist. Wenn es breiter als hoch ist, werden die enthaltenen
    Rechtecke nebeneinander, sonst übereinander platziert.'''
    if self.breite >= 60 and self.hoehe >= 60:
      if self.breite > self.hoehe:
        x = random.uniform(self.links+30, self.rechts-30)
        r1 = Rechteck(self.links+10, self.oben-10, x-5, self.unten+10)
        r2 = Rechteck(x+5, self.oben-10, self.rechts-10, self.unten+10)
        self.inhalt = [r1,r2]
      else:
        x = random.uniform(self.oben-30,self.unten+30)
        r1 = Rechteck(self.links+10, self.oben-10, self.rechts-10, x+5)
        r2 = Rechteck(self.links+10, x-5, self.rechts-10, self.unten+10)
        self.inhalt = [r1,r2]
    return self.inhalt

  def loesche(self):
    '''Das Rechteck wird mit seinem ganzen Inhalt, also rekursiv,
    aus der Liste mit allen Rechtecken gelöscht. Die Rechtecke werden
    von innen nach außen mit der Hintergrundfarbe übermalt.'''
    global alle_rechtecke
    alle_rechtecke.remove(self)
    for irechteck in self.inhalt:
      irechteck.loesche()
    stift.color(BGCOLOR)
    self.zeichne()


def schluss(x,y):
  '''Das Turtle-Fenster wird geschlossen.'''
  global screen
  screen.bye()

def setup_turtle():
  '''Turtle(-Stift) und Turtle-Fenster werden vorbereitet.'''
  global stift, screen
  stift=turtle.Turtle()
  screen=stift.getscreen()
  screen.title('Rechteckschoner')
  screen.bgcolor(BGCOLOR)
  screen.setup(SCREEN_WIDTH,SCREEN_HEIGHT,None,None)
  screen.listen()
  screen.onclick(schluss) # Mit einem Klick die Turtle beenden.
  stift.reset()
  stift.speed(SPEED)

def zeichne_rechtecke(ausseneck):
  '''Ein Rechteck wird mit seinen inneren Rechtecken gezeichnet.'''
  aktuelle_rechtecke = []
  aktuelle_rechtecke.append(ausseneck)
  # Alle Rechtecke werden gezeichnet, so lange Platz da ist.
  for r0 in aktuelle_rechtecke:
    for r in r0.fuelle():
      aktuelle_rechtecke.append(r)
    r0.zeichne()
  return aktuelle_rechtecke

def main():
  global alle_rechtecke
  setup_turtle()
  alle_rechtecke = []
  # Mit dem ersten, äußersten Rechteck geht es los.
  starteck = Rechteck(-SCREEN_WIDTH/2+20,SCREEN_HEIGHT/2-20,
            SCREEN_WIDTH/2-20,-SCREEN_HEIGHT/2+20)
  # Endlosschleife, in der Rechtecke gezeichnet und gelöscht werden.
  while 1:
    alle_rechtecke.extend(zeichne_rechtecke(starteck))
    # Ein Rechteck zum Löschen auswählen und löschen ...
    wegeck = random.choice(alle_rechtecke)
    wegeck.loesche()
    # ... und mit neuer Farbe zum neuen Starteck machen.
    wegeck.waehle_farbe()
    starteck = wegeck
  return "EVENTLOOP"

if __name__ == '__main__':
  msg = main()
  print(msg)
  mainloop()
```