py2cd - Zeichnen mit Python Teil II

1.1 Vorbereitung

Um diese Tutorial erfolgreich abzuschließen, muss zuerst pygame und py2cd installiert sein. Eine Installationsanleitung ist unter `https://github.com/coderdojoka/py2cd/` zu finden. Desweiteren sollte Teil I dieser Tutorial-Serie verstanden sein.

1.2 Einleitung

1.2.1 Animationen - Bilder pro Sekunde

Um Bilder zu flüssigen Animationen/Videos zusammenzufügen, werden mehrere Bilder schnell hintereinander angezeigt. Ein Film im Fernsehen wird mit ca. 24 Bilder pro Sekunde abgespielt und das Auge nimmt diese als Film war.
In Computerspielen ist dies ähnlich. Der Spielinhalt wird mehrmals pro Sekunde neugezeichnet. Die Bildwiederholungsrate wird dabei als 'Frames per second' (fps) = 'Bilder pro Sekunde' (bps). Eine Bildwiederholungsrate von 30fps bedeutet also, dass der Bildschirminhalt 30 mal pro Sekunde neugezeichnet wird!

1.2.2 Die Aktualisierungsschleife

Standartmäßig wird der Bildinhalt mit 30fps neugezeichnet. Jedes Mal wenn dies geschieht, wird eine definierte Aktualisierungsfunktion ausgeführt. Hier ein komplettes Beispiel mit Aktualisierungsfunktion:

```
__author__ = 'Mark Weinreuter'

# Diese zwei Zeilen werden immer benötigt, um py2cd zu
 ↪  importieren
from py2cd import *
from py2cd.farben import *

def aktualisiere(delta):
    print("Wird ca. 30 mal pro Sekunde ausgeführt.")
    # Die Box um 1 pixel verschieben
    box.aendere_position(1, 1)


# Initialisiert das Fenster
Spiel.init(400, 400, "Bewegung", aktualisiere)

# So kann man die Aktualisierungsfunktion ändern
Spiel.setze_aktualisierung(aktualisiere)
```

```
box = Rechteck(10, 10, 50, 50, ROT)

# Das Spiel starten
Spiel.starten()
```

Die Aktualisierungsfunktion ist wie der Name sagt eine Funktion. Diese benötigt genau einen Parameter:
Dieser Parameter kann zunächst einfach ignoriert werden. Eine Erklärung findet sich am Ende dieses Tutorials.
Um die Aktualisierungsfunktion zu setzen gibt es 2 Möglichkeiten:

```
# 1.) in Spiel.init(..., aktualisiere)
Spiel.init(400, 400, "Steuere das Rechteck!", aktualisiere)

# 2.) Mittels Spiel.setze_aktualisierung
Spiel.setze_aktualisierung(aktualisiere)
```

Diese Funktion wird automatisch von py2cd mit der gewünschten fps-Rate aufgerufen.

1.3 Bewegung

Es gibt verschiedene Möglichkeiten ein Objekt zu bewegen. Wie zuvor beschrieben ist jedes Objekt über eine x- und y-Koordinate im Raum verankert. Man kann ein Objekt, und damit dessen Koordinaten mit folgendne Funktionen bewegen:

1.3.1 Relative Änderung

```
box = Rechteck(10, 10, 50, 50, ROT)
# Ändert die x- und y-Koordinate um 5 Pixel
box.aendere_position(5, 5)
```

1.3.2 Absolute Position setzen

```
box = Rechteck(10, 10, 50, 50, ROT)
# Setzt die x- und y-Koordinate auf die gegebenen Werte
box.setze_position(50, 50)
```

1.3.3 Geschwindigkeit festlegen und bewegen

```
box = Rechteck(10, 10, 50, 50, ROT)
# Die x- und y-Geschwindigkeit setzen
```

```
box.setze_geschwindigkeit(6,6)
# Bewegt die Box bei jedem Aufruf von bewege() um 6 Pixel
box.bewege()
```

Diese Bewegungsmöglichkeiten sind ähnliche wie die in Scratch. `aender_position` verhält sich genau wie `ändere x um ...` und `aender y um ....`
`setze_position` verhält sich genau wie `setze x auf ...` und `setze y auf ....`

Erinnerung: Der Koordinatenursprung liegt in der linken oberen Ecke. Eine Änderung der x-Koordinate, die größer 0 ist, ist eine Bewegung nach rechts. Eine negative Änderung bedeutet eine Bewegung nach links.
Eine Änderung der y-Koordinate, die größer 0 ist, ist eine Bewegung nach **unten!**. Dies ist zunächst verwirrend, ist allerdings bei vielen Computerprogrammen so. Eine negative Änderung bedeutet demzufolge eine Bewegung nach oben!

```
# Ändert die x- und y-Koordinate um 5 Pixel
box.aendere_position(5, 5)
```

Dies ist also eine Bewegung um jeweils 5 Pixel nach rechts unten!

1.3.4 Position abrufen

Man kann die x- und y-Koordinaten eines Objektes ganz einfach abfragen

```
# Box an der Stelle 10x10
box = Rechteck(10, 10, 50, 50, ROT)
# x- und y-Koordinate abfragen und ausgeben
print(box.x, box.y)
```

1.4 Aufgaben

1. Bewege ein Objekt mit den verschiedenen Funktionen über die Spielfläche. Kannst du verhindern, dass ein Objekt die Spielfeldgrenzen verlässt.
   Hinweis: Kannst du dir eine if-Schleife ausdenken, die die Position des Objekts überprüft?

1.5 Anhang

1.5.1 Erklärung zum Delta-Parameter der Aktualisierungsfunktion

Dieser Parameter gibt an, wieviel Zeit vergangen ist relativ zu der gewünschten fps-Rate. Z.B. Mit 30fps, wird das Bild 30mal pro Sekunde gezeichnet. Benötigt ein Zeichenvorgang

genau 1/30 Sekunde, so ist der Wert: `delta = 1`.
Dauert der Zeichenvorgang länger als 1/30s so ist `delta > 1`. Dementsprechend ist `delta < 1`, falls der Zeichenvorgang schneller als 1/30 s war.
Die Aktualisierungfunktion wird nicht immer exakt mit der gewünschten Update-Rate ausgeführt. Um diesen Effekt entgegen zu wirken können Positionsänderungen mit diesem Wert multipliziert werden. Z.B: so:

```
box.aendere_position(1 * delta, 1 * delta)
```

Dies ist allerdings ein sehr fortgeschrittenes Verfahren und kann ignoriert werden. Es wird nur der Vollständigkeit wegen erklärt. :)