py2cd - Bilder und Animationen

Ein gutes Spiel zeichnet sich durch coole Grafiken und Animationen aus. Mit den vorhanden Mittel ist es jedoch ein bisschen umständlich alles selber aus Rechtecken, Kreisen, etc... selbst zu zeichnen.
Deswegen ist es eine gute Idee Bilder in Bildbearbeitungsprogrammen zu Malen oder von anderen Malen zu lassen :). Diese Bilder können dann ganz einfach geladen und angezeigt werden.

1.1 Ein Bild laden und anzeigen

Um ein Bild zu laden gibt es den *BilderSpeicher*. Dieser verwaltet alle Bilder, die du benötigst. Du musst Bilder einmal laden und kannst sie dir beliebig oft wieder holen und anzeigen lassen. Damit du dein Bild später wiederfindest, musst du jedem Bild einen eindeutigen Namen geben:

```
# Das Bild in den BildSpeicher laden und ihm einen Namen geben:
 ↪ 'scratch'
BildSpeicher.lade_bild("scratch", "testimages/scratch.png")

# Das Bild über seinen Namen holen und anzeigen
bild1 = BildSpeicher.gib_bild("scratch")
bild1.setze_position(250, 100)

# Das Bild (nocheinmal) über seinen Namen holen und anzeigen
bild2 = BildSpeicher.gib_bild("scratch")
bild2.setze_position(200, 300)
```

1.1.1 Woher weiß mein Computer welches Bild geladen werden soll?

Das Bild, das angezeigt werden soll muss auf deinem Computer gespeichert sein. Dann muss man den Dateinamen und Pfad, d.h. alle Ordner- und Unterordnernamen, dorthin angeben.

> Ein Dateipfad kann auf 2 Arten angeben werden. Als **relativer** Pfad oder als **absoluter** Pfad.
> Ein relativer Pfad bedeutet, dass man den Pfad von dem Ordner aus angibt, in dem auch dein Programm liegt. Gibt man also den Dateipfad als 'scratch.png' an, so sucht Python nach der Datei mit dem Namen 'scratch.png', im dem Ordner, in dem auch dein Programm liegt.
> Ein absoluter Pfad beginnt immer auf unterster Ebene des Dateisystems. D.h. willst du eine Datei auf dem Desktop laden, könnte der Pfad unter Windows z.B. so lauten: *'C:\Users\Mark\Desktop\scratch.png'*.

Das Einfachste ist, wenn du Bilder immer dem Ordner speicherst, indem dein Programm liegt. Du kannst sie natürlich auch in einen Unterorder z.B. 'Bilder' in deinem Programm-Ordner speichern und den Pfad angeben als: *'Bilder/scratch.png'*

1.1.2 Bilder skalieren und rotieren

Man kann von Bilder und Animationen ganz leicht die Größe ändern und diese drehen. Bilder werden dabei immer um ihren Mittelpunkt gedreht, und auch beim Skalieren bleibt der Mittelpunkt des Bildes fest.

```
bild1.setze_rotation(180)  # um 180, d.h. auf den Kopf stellen

bild2.aendere_skalierung(.5)  # Ändert die aktuelle! Skalierung
 ↪ um die Hälfte
bild2.aendere_rotation(90)  # Die aktuelle! Rotation um 90

bild1.rotiere_und_skaliere(180, 2.0)  # Dreht um 180 und
 ↪ skaliert um das 2-fache
bild1.aendere_rotation_und_skalierung(90, 1.2)  # ändert
 ↪ Rotation und Skalierung
```

1.2 Animation laden und anzeigen

Eine Animation ist eigentlich nur eine Folge von Bilder, die schnell hintereinander durchgewechselt werden. Dementsprechend benötigt man für eine Animation eine Liste von Bilder und Zeitabständen, in denen die Bilder gewechselt werden sollen.

```
# Bilder können über ihren BilderSpeicher-Namen geladen werden
# oder über den Dateipfad!
bilder_liste = [
    "testimages/n1.png",  # über den Dateipfad laden
    "testimages/n2.png",
    "testimages/n3.png",
    "testimages/n4.png",
    "testimages/n5.png",
    "testimages/n6.png",
    "testimages/n7.png"
]

# Eine neue Animation mit unseren Bilder, die endlos wiederholt
 ↪ wird
```

```
animation = BildAnimation(bilder_liste, True)
# rechts unten im Eck positionieren
animation.rechts = 20
animation.unten = 20

# Die Animation starten
animation.start()
```

1.2.1 Eine Animation steuern

Eine Animation kann gestartet, pausiert und gestoppt werden. Diese Funktionen verhalten sich, wie man erwartet, so wird z.B. die Animation nach einer Pause wieder an der pausierten Stelle fortgesetzt.

```
animation.pause()  # pausieren, .start() macht an der aktuellen
 ↪ Stelle weiter
animation.stop()  # komplett anhalten und verstecken, .start()
 ↪ beginnt von vorne.
animation.start()  # neustarten oder fortsezten

# ein bestimmtes Bild, hier das 3.te (wir zählen ab 0!)
 ↪ anzeigen
# Hält die Animation an! Wie .stop(), zeigt das ausgewählte
 ↪ Bild an
animation.zeige_bild(2)
```